PRIORITY BASED SWITCH ALLOCATOR IN ADAPTIVE PHYSICAL CHANNEL REGULATOR FOR ON CHIP INTERCONNECTS

A Thesis

by

SONALI MAHAPATRA

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Eun Jung Kim
Co-Chair of Committee,	Anxiao Jiang
Committee Member,	Jiang Hu
Head of Department,	Nancy M. Amato

August 2014

Major Subject: Computer Engineering

ABSTRACT

Chip multiprocessors (CMPs) are now popular design paradigm for microprocessors due to their power, performance and complexity advantages where a number of relatively simple cores are integrated on a single die. On chip interconnection network (NoC) is an excellent architectural paradigm which offers a stable and generalized communication platform for large scale of chip multiprocessors. The existing model APCR has three regulation schemes designed at switch allocation stage of NoC router pipelining, such as monopolizing, fair-sharing and channel-stealing. Its aim is to fairly allocate physical bandwidth in the form of flit level transmission unit while breaking the conventional assumptions i.e. flits size is same as phit size. They have implemented channel-stealing scheme using the existing round-robin scheduler which is a well known scheduling algorithm for providing fairness, which is not an optimal solution.

In this thesis, we have extended the efficiency of APCR model and propose three efficient scheduling policies for the channel stealing scheme in order to provide better quality of service (QoS). Our work can be divided into three parts. In the first part, we implemented ratio based scheduling technique in which we keep track of average number of flits sent from each input in every cycle. It not only provides fairness among virtual channels (VCs), but also increases the saturation throughput of the network. In the second part, we have implemented an age based scheduling technique where we prioritize the VC, based on the age of the requesting flits. The age of each request is calculated as the difference between the time of injection and the current simulation time. Age based scheduler minimizes the packet latency. In the last part, we implemented a Static-Priority based scheduler. In this case, we arbitrarily assign

random priorities to the packets at the time of their injection into the network. In this case, the high priority packets can be forwarded to any of the VCs, whereas the low priority packets can be forwarded to a limited number of VCs. So, basically Static-Priority based scheduler limits the accessibility on the number of VCs depending upon the packet priority.

We study the performance metrics such as the average packet latency, and saturation throughput resulted by all the three new scheduling techniques. We demonstrate our simulation results for all three scheduling policies i.e. bit complement, transpose and uniform random considering from very low (no load) to high load injection rates. We evaluate the performance improvement because of our proposed scheduling techniques in APCR comparing with the performance of basic NoC design. The performance is also compared with the results found in monopolizing, fair-sharing and round-robin schemes for channel-stealing of APCR. It is observed from the simulation results using our detailed cycle-accurate simulator that our new scheduling policies implemented in APCR model improves the network throughput by 10% in case of synthetic workloads, compared with the existing round-robin scheme. Also, our scheduling policy in APCR model outperforms the baseline router by 28X under synthetic workloads.

To my parents

ACKNOWLEDGEMENTS

I would like to thank my advisor Professor Eun Jung (EJ) Kim for her invaluable and continuous support in making this work possible. I thank her for the opportunity and freedom she gave me for working on an entirely new topic within her research group. A special thanks to Dr. Anxiao Jiang and Dr. Jiang Hu for serving on my committee.

I would like to thank my loved ones, who have supported me throughout the entire process.

NOMENCLATURE

NoC	Network on chip
VC	Virtual channel
APCR	Adaptive physical channel regulator
QoS	Quality of service
PF	Proportional fair
RR	Round-robin
CMP	Chip multiprocessors
PR	Priority
SA	Switch allocation
VA	VC allocation
IP	Input port
BC	Bit complement
TP	Transpose
UR	Uniform random

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

Chip Multiprocessor [1] is a single integrated circuit silicon chip which is embedded with a large number of very simple cores and memory components. It holds the vision of translating Moore's law into constant performance improvement by integrating large number of cores on a single chip. The performance is scaled by incorporation of large number of cores on the single die and utilizing higher levels of thread level parallelism (TLP). Complexity is eliminated by designing relatively simple cores on the die. CMPs are vastly used in many application domains, such as general purpose, graphics and embedded etc. CMPs were initially designed in early 2000s by Intel and other manufactures. One of the examples of multi-core systems is Intel Polaris with 80 cores. As CMPs being popular in designing modern micro processors, provision of an efficient communication method becomes a necessity for large scale chip multiprocessors. Moreover, there is diversity in the size of packets can be found in CMP communications which also restricts proper utilization of wide bandwidth physical links. Conventional shared buses and dedicated wires could not satisfy the requirements of on chip communication for future multi-core architectures.

In this regime, on chip interconnection network (NoC) is an outstanding architecture that offers a stable and generalized communication platform for large scale chip multiprocessors. As the on-chip network size continues to increase, the bandwidth required to support concurrent computations on all cores increases in order of magnitude. Hence, NoC should be carefully designed to meet the high bandwidth requirement of future CMPs. Wires are becoming abundant resources available in NoC with its shrinking feature size [2, 3, 4]. Therefore, the huge wiring capability leads to wide physical channels among routers that facilitates low latency due

to small serialization delay. Also, there is limited buffer area budget in NoC that proportionately decreases the VC depth with increasing flit size. This causes more contention in the network which eventually leads to performance degradation of the system. In order to eliminate this problem, it is necessary to introduce smaller flow control units to achieve finer granularity in communication. Since NoC can benefit from these abundant wiring resources available on chip, it is very crucial to find a way that can fully utilizes these resources.

1.1 Motivation

Allocators are particularly important aspects of router design, as they directly affect overall network performance in several ways. For example, physical link bandwidth utilization, and crossbar port utilization can be affected by the quality of arbitration we use. Again, the quality of allocation is determined in the order of matching sets between requests and available resources. So, switch allocation has immediate impact on the network's overall throughput under different workload of various traffic patterns. It is because, the queuing delay that packets incurred in a congested network during this phase. Furthermore, allocators control the network's fairness properties. Finally, allocator directly affects the critical path delay in many typical router designs. Consequently, delay-optimized allocator implementations are required so that the network may able to achieve high processing frequency. Also, as wires are being abundant resources available on a single die in NoC, it is very crucial to fully utilize these resources to provide high performance. It is observed that, if we simply increase flit size to be the same as the phit size, we cannot achieve significant performance improvement for the same router buffer budget. Again, diverse packet sizes are also introduced in CMP architecture which leads to a complicated credit management. So the domain of our work is based on modifying the SA stage of

basic NoC that is described in [5]. Previously, several works in NoC architectures have tried to satisfy quality of service requirement which is needful in case of running real time applications. We need a method that can achieve quality of service requirements at execution time with less computations.

The motivation behind our work is to make the APCR model more efficient by overcoming some of the drawbacks that exists in round-robin scheme implemented in [5]. The channel-stealing scheme in APCR model is observed to be performing the best among all the proposed regulation schemes. The scheduling at SA stage in channel-stealing scheme is implemented using round-robin policy. Generally, it chooses the requesting VCs in cyclic order providing a fairness in selection procedure. However, this scheduling technique is not efficient enough to satisfy a number of quality of service requirements, such as minimizing packet latency, providing support for diverse applications, or handling vulnerable packets depending upon their degree of criticality. For example, if VC_0 is scheduled in cycle 0, then according to round-robin policy, next selected VC would be VC_1 (say) and so on. If all VCs have requests to same output path, then the chances of getting scheduled again for VC_0 would be the least in next cycle. In this situation, the latency of the packet residing in the VC_0 will keep increasing. This technique performs even worse if the packet residing in VC_0 are considered to be time critical as the chances of missing their deadline are more. So to overcome this problem, we need to implement some deadline sensitive scheduling policy in order to handle both criticality and reduce packet latencies. Also, NoC infrastructure should have the ability to guarantee timely transfer of packets for real time applications. It needs to provide diversity to support different levels of applications. If a scheduling policy can handle critical packets, we can also be able to provide a provision for diverse application to be handled in the APCR. So, all these factors motivate us to implement new scheduling algorithms which would also

be responsible for maximizing throughput and satisfying the quality of service.

1.2 Contribution

In this thesis, we have implemented three scheduling policies for the channel-stealing scheme in APCR model [5]. The three scheduling policies aim to provide different QoS for different applications. The main contributions of this thesis are:

1) We implement Proportional fair scheduling policy for the channel-stealing scheme in APCR model. It has been proved that PF policy provides fairness to the nodes and maximizes average system throughput, therefore using this policy we aim to serve the input ports and VCs during switch arbitration stage in NoC with the above mentioned features.

2) We design an Age based switch allocation algorithm for the channel-stealing scheme in APCR model to provide minimum packet latency as a QoS.

3) We design a Static-Priority based scheduling policy to prioritize different packets. It is done so to serve different applications in APCR model with different QoS.

4) Finally, we present the simulation results and performance difference between our proposed and existing techniques to validate our three proposed algorithms.

1.3 Organization

The rest of the thesis is organized as follows. We briefly explain the NoC concept and some of its important terminologies in Section 2. we present the concepts and working models of APCR router in Section 3. We summarize the related works in this area in Section 4. In Section 5, we present our proposed scheduling policies with prior study and demonstration of our algorithm. In Section 6, we explain the design methodology and simulation results. In the end, we conclude our work in Section 6.

2. NETWORK ON CHIP

NoC can be defined as an on chip interconnection design embedded on a single chip like a micro network that consists of large number of cores where, each core is connected to a router via network interface. On chip communication architectures provide outstanding platform to design future system on chip (SoC). Its model is outlined with scalable components to optimize network-level performance measures, such as average packet delay or saturation throughput that overcome certain design problems in traditional SoCs. As technology scales down, the number of processing elements integrated on a single die increases rapidly. So, designing NoC becomes an important concern in chip multiprocessor architectures. A typical multi core network on chip structure is presented in figure 2.1. Routers are connected to each other through physical links which are available in large scale. In order to achieve high throughput with low delay, it is also crucial to utilize each available resources, such as clock cycle, area etc very efficiently. Important features in NoC include network interface (NI) which is responsible for packetization, virtual channels (VC) which are responsible for transmission of packets, and the physical links that are responsible for packet traversal throughout the network. NoCs are intended to optimize system level performance measurements, such as system throughput and as well as average packet latency. An efficient NoC design can be characterized by several key design choices, such as: network topology, routing policy, network protocol, router pipelining etc which are explained briefly in next section. In our work, we have considered 64 cores $2D$ Mesh topology, XY routing with wormhole switching as our design standard. The $2D$ Mesh topology is a practical and wide spread topology in which each router is connected to each other via physical wires which are bidirectional in

nature. Again, each router attaches the core via network interface (NI) where the cores communicates with each other via router and routing algorithms. The core is a processing element where the control unit resides and the computation is performed and it regulates all the activities on chip. The unit of communication is a packet which comprises of a number of flits, in our case one packet equals four flits. Among these four flits, the head flit contains all the routing information and also about the destination router information for the corresponding packet. In wormhole switching, the packets can be split up into flits and they can traverse the network in a pipe-lined manner. The switching requires only one buffer to forward flits from upstream to down stream router. Again, XY routing is otherwise knows as dimension order routing which is very simple form of routing a network and follows a deadlock free routing policy where each packet first routes on X-direction and then on the Y-direction to reach its destination.

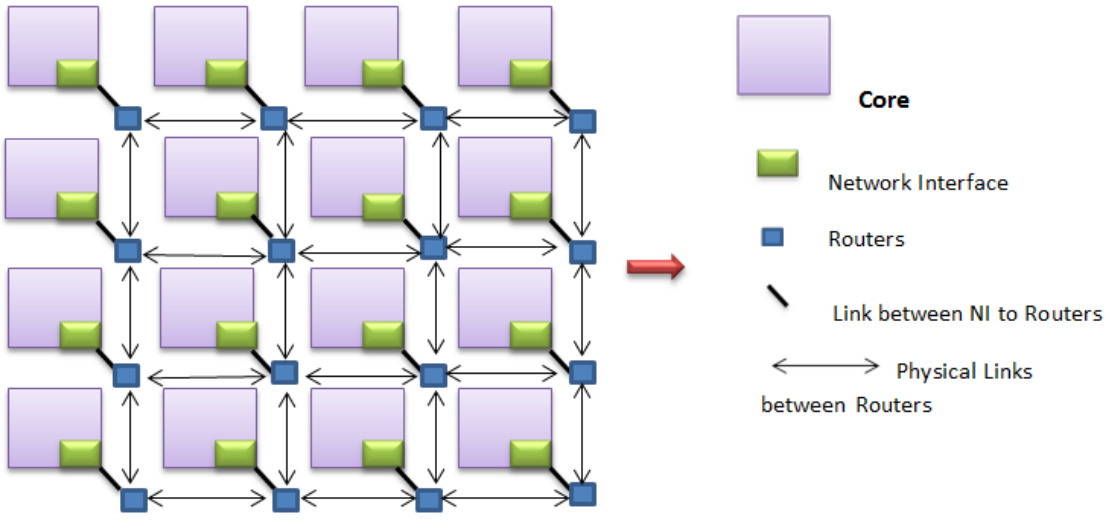

Figure 2.1: Basic 8x8 2D mesh NoC topology

For optimal utilization of the resources we discussed before each NoC router works in pipelining manner which is a very important concept and is discussed in next section. There are five important stages, such as router computation (RC), VC allocation (VA), switch allocation (SA), switch traverse (ST) and link traverse (LT) on order to process multiple packets at same clock cycle. Once a packet has completed VC allocation, its flits can be forwarded to the selected destination port depending upon the buffer space availability. There is a crossbar connection which is established between I/O ports of the switch whenever there is request from packets to traverse to next router through that switch. The switch allocator is responsible for scheduling this crossbar mapping; in particular, it generates matching between VC requests at input port of switch to its output ports. As various VCs from different input ports will compete for the same output path at the crossbar and also congestion must be avoided, hence the arbitration stage becomes a critical stage for the performance improvement. The quality of the generated matching directly affects the packet latency and throughput of the network under given work load which in particular needs a good scheduling algorithm. So this is the main motivation behind our work. We will be presenting more about it in next section of the paper. At present, NoCs are considered as a promising method for dealing with the communication demands in case of large scale chip multiprocessors. Such packet-switched on chip interconnect paradigm consists of a set of routers that are communicated to each other, which are characterized by a set of following basic parameters.

2.1 Topology

The NoC Topology is responsible for dictating the number of routers and channels and the connectivity among them. It sets up basic limits for overall network performance and energy efficiency by estimating network diameter and bisection

bandwidth. Moreover, the NoC topology is meant to be a critical factor in determining overall network cost as it controls the number and the size of individual network components of NoC. As Mesh is very simple yet a regular topology mostly used in every research in NoC, we have considered Mesh as the standard topology in our work. In Mesh topology, each router is connected via a bidirectional physical channel to its four neighboring routers. There are new outstanding topologies that have been introduced, such as concentrated Mesh (MESH), flattened butterfly (FBLY), MeshX2, MeshX4 etc which surely make very good improvement in performance of NoC.

2.2 Routing

The second factor is computing path from source to destination using appropriate routing functions. The routing function determines the path that a packet must take to accomplish its task from source to destination. A good routing also takes care of avoiding deadlock while routing from hop to hop inside a network. So, it affects the average hop count and the degree of load balance across network channels. There are two important constraints, such as delay and cost, because of which, NoCs exploit simple arithmetic routing functions like dimension order routing. In this case, the next hop is computed depending upon the address of current router and the destination address. In our domain, XY-routing is being considered as a standard routing algorithm where the next hop of the packet is calculated depending upon the deadlock avoidance policies in which the packet must go first X-direction and then Y-direction.

2.3 Flow Control

The flow control monitors the network resources like buffer capacity, physical channel bandwidth and credits in each virtual channel etc. A credit based flow

control considers a credit factor which is tracked by current upstream router for the credit of down stream router. The number of available free slots in next router's buffer is considered as the number of credits. So, when a down stream router transmits flits to the next router, its credit is increased which is maintained at its corresponding upstream router. It also takes care of control units of each packet. Eventually, flow control regulates resource utilization and hence flow control is a significant factor in affecting the network performance. Our work employs virtual channel (VC) flow control. Generally, the control unit is called a phit which is of same size as that of transmission unit which is known as flit. Each virtual channel can keep maximum four flits at a time.

2.4 NoC Router Pipelining

The router in NoC is solely responsible to process each packet that arrives at its input ports and forward it to appropriate correct output VC in next router. In order to utilize the resources optimally and process multiple packets at each cycle, the router goes through several stages that are demonstrated in figure 2.2.

Figure 2.2 represents a basic architecture of NoC Router and its pipeline stages. In the figure the components used for each stage is also shown. There are four input ports, each consists of four VCs and each VC consists of four buffers to store the incoming packets. Each VC is 4 flits long. There are generally five pipeline stages in an NoC router, such as route computation, VC allocation, switch allocation, switch traverse and finally link traverse.

We consider the size of packet to be of four flits among which there is one head flit that keeps all routing information, two middle flits and one tail flit. As the packet arrives at the router's input port, routing computation is performed and next hop is determined. Then, VC allocation was done to reserve the VC of next hop before it

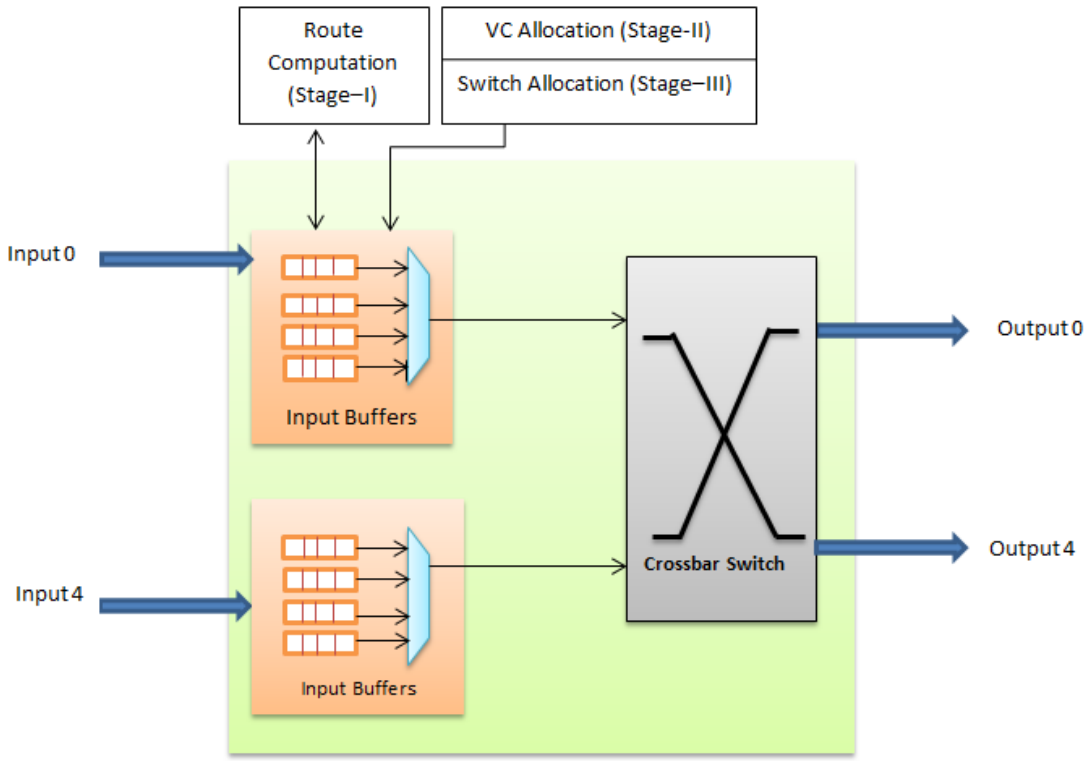

Figure 2.2: Basic router architecture

can be forwarded to its output port. Then switch allocation is performed to reserve crossbar ports in order to traverse the crossbar before it can be forwarded to the physical link. Then the flits traverse the link in link traverse stage and reach at next router's VC. We have briefly described the basic task that are performed at each stage below.

2.4.1 Routing Computation (RC Stage)

As we mentioned before, the routing computation is done by different methods, such as XY routing, and dimension order routing etc. In this stage, the routing computation logic determines the appropriate output path. This step is only accomplished at head flit of each router. Then, the subsequent three flits can be forwarded at same flow path. As latency is being a critical performance metric in many CMPs, NoC routers commonly implement look ahead routing in order to reduce pipeline delay.

2.4.2 VC Allocation (VA Stage)

After successful computation of next hop of the packet, the packet needs to have exclusive access to VC of next selected router. Again, this stage can only be performed on head flit of the packet. So, the component used in this case is called VC allocator which assigns available output VCs to waiting packets at the router's input ports. Then, rest of the flits follows the head flit.

2.4.3 Switch Arbitration (SA Stage)

After successful computation of VC at next router, the packet participates in switch allocation stage. The component used here is known as switch allocator which establishes a schedule for each flits of every packet before it can transmit through the physical wire. It assigns the crossbar ports to each flits of the packet because,

unlike other previous stages, this stage is performed on each and every flits of the packet. Switch allocation is being a vital stage in reducing overall packet latency, so we need an optimal scheduling policy to provide best service that also can achieve fairness. There are different arbitration schemes, such as hierarchical round-robin, fast come fast serve etc.

2.4.4 Link Traversal (LT Stage)

Finally, after successful assignment of switch ports, the flits are ready to traverse through the physical link to the next hop in the last cycle. This will be the last stage of NoC router pipelining. So, like in switch allocation (SA) stage, link traversal is also performed on each individual flits of the packet.

3. ADAPTIVE PHYSICAL CHANNEL REGULATOR

3.1 Introduction

Paper [5] has introduced an adaptive Physical Channel Regulator (APCR) in which the NoC baseline router is reimplemented with additional features in between VC and switch arbitration stage. The new features can be explained in terms of three regulation schemes such as monopolizing, fair-sharing and channel-stealing. They have considered the transmission unit (flit) size to be less than the control unit (phit) size to achieve finer granularity in flow control. That means phit size to be four times of flit size. Also, the APCR router permits VCs to transmit multiple flits in same cycle. They have also followed credit based flow control mechanism where the upstream router tracks the number of free buffers at the receiving routers. Every time, the upstream router sends a flit to its downstream router, credit of the downstream router is being deducted by one until its credit counter becomes zero. That means all VCs at the downstream router are full if the credit becomes zero for any router and thus it is unable to receive any flits from its upstream routers. On the other hand, when downstream router sends a flit to its corresponding downstream router, making its associated VC one buffer free, the credit of the router increases by one.

The figure 3.1 represents the proposed APCR router structure in [5]. In the figure, the APCR component works in switch allocation stage. In this APCR router, working principle of all pipeline stages are same as the basic NoC router, with only difference in the SA stage. The APCR component works along with switch allocator and virtual channels. The three regulation schemes intelligently assign the output channel resources considering the dynamic status of physical channel's availability

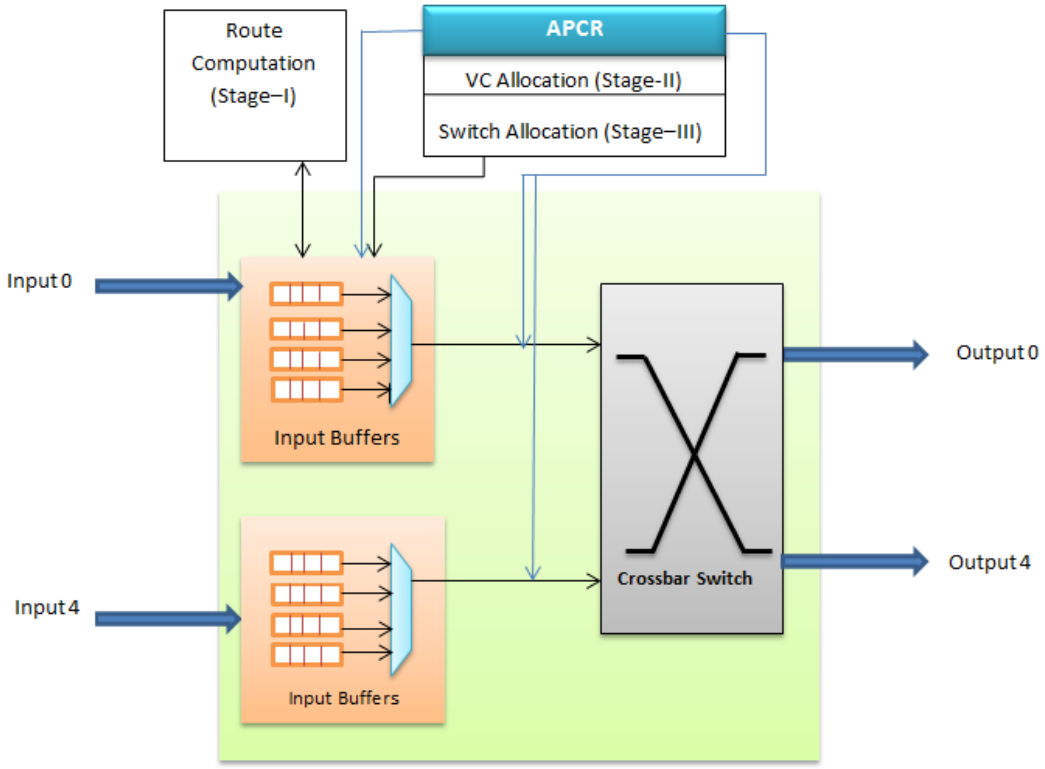

Figure 3.1: Structure of an NoC router. Reprinted with permission from [5]

and input buffers occupancy. First, it selects the features of appropriate regulation schemes choosing any one among above three schemes. Then, it finalizes the number of flits that each virtual channel is allowed to send according to the scheme. Each VC can send a variable number of flits in each cycle depending on the current network status. For example, in monopolizing scheme, only a single VC can be selected at SA stage to send multiple flits belong to same packet. In fair-sharing scheme, a VC is assigned dedicated sub channels at each output port. So, one VC can send one flit each through its dedicated sub channel to the network. In fair-sharing, the wide physical channel bandwidth is partitioned into four reserved sub channels for each individual VC at each input port. The only drawback on this scheme is that if the

VC corresponding to a particular sub-channel has no flit to send in the cycle, then that sub channel bandwidth remains unused.

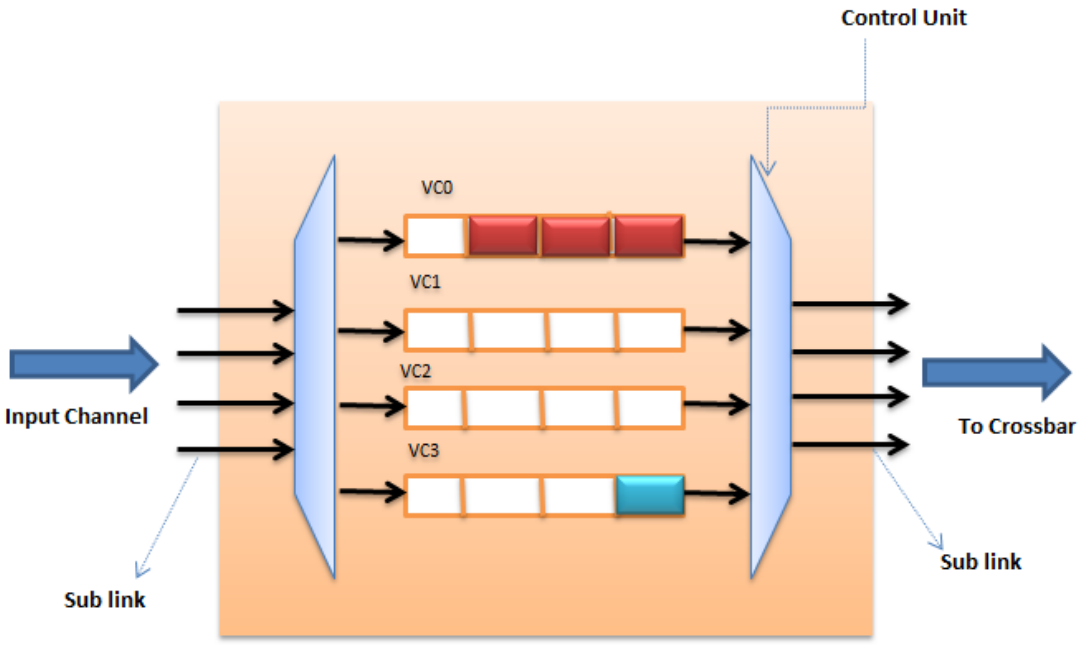

Figure 3.2: Overview of Channel-stealing mechanism. Reprinted with permission from [5]

Figure 3.2 shows the situation where channel-stealing was introduced in order to solves the problem in fair-sharing. It allows VC from same or different input ports to use any sub channel at the output port if they have valid requests to the same out put direction. Channel-stealing exploits the same arrangement as fair-sharing besides the fact that one sub channel can be utilized by any VCs in the router with valid requests if the owner VC for that sub channel does not have any flits to send flits in that cycle.

Figure 3.3 presents the buffer management in APCR router. There are three

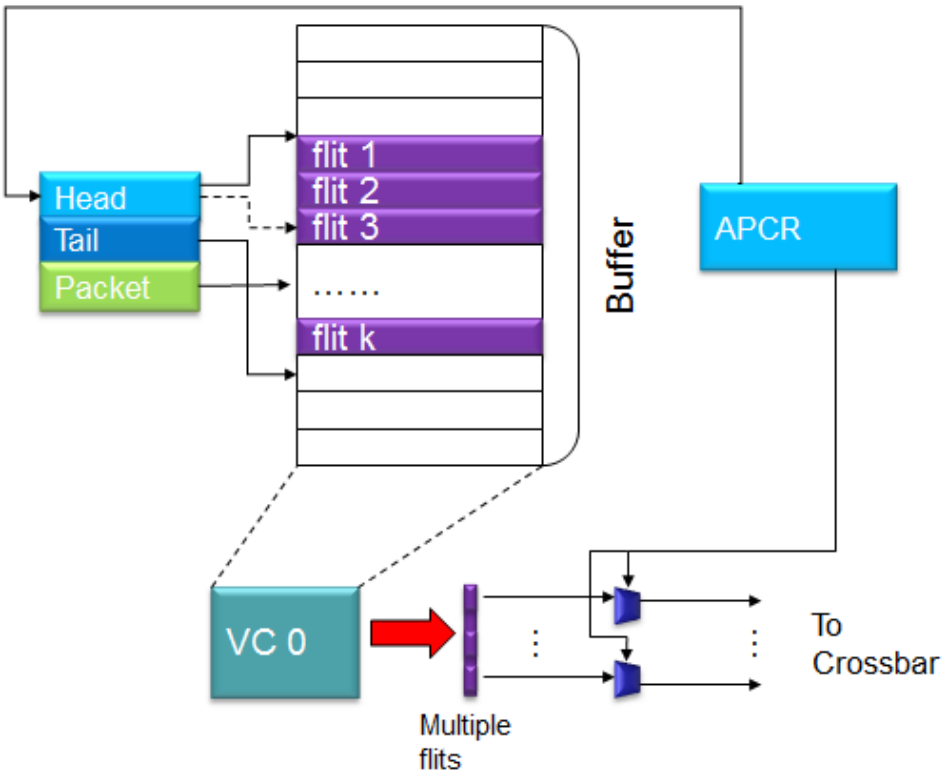

Figure 3.3: Overview of buffer management. Reprinted with permission from [5]

steps in buffer management in order to send multiple or single flits depending upon any of the three regulation schemes. In first step, multiple flits are read out, because the width of buffer output and the router output channel is same where the output channel width is a multiple of flit size. APCR keeps track of the number of flits each VC can send. In second stage, the head pointer of each input buffer is controlled by the allocation information provided in previous step. In third stage, the actual number of flits that are guaranteed to be sent in that cycle are forwarded to downstream router using the crossbar and remaining flits read in first stage will be canceled and again read out from same VC in the next cycle.

3.2 Channel-stealing in APCR

As we mentioned before, APCR router has been implemented over baseline router by modifying the SA stage in order to achieve better performance. In our work, we emphasize on improving scheduling policy in channel-stealing scheme of APCR. Monopolizing has two stage switch allocation same as in basic router. Moreover, it permits only single VC to utilize the whole output bandwidth which leads to inefficient utilization of the physical bandwidth. At the same time, the fair-sharing dedicates each sub channel to corresponding input VC and thus eliminates the first stage switch allocation.

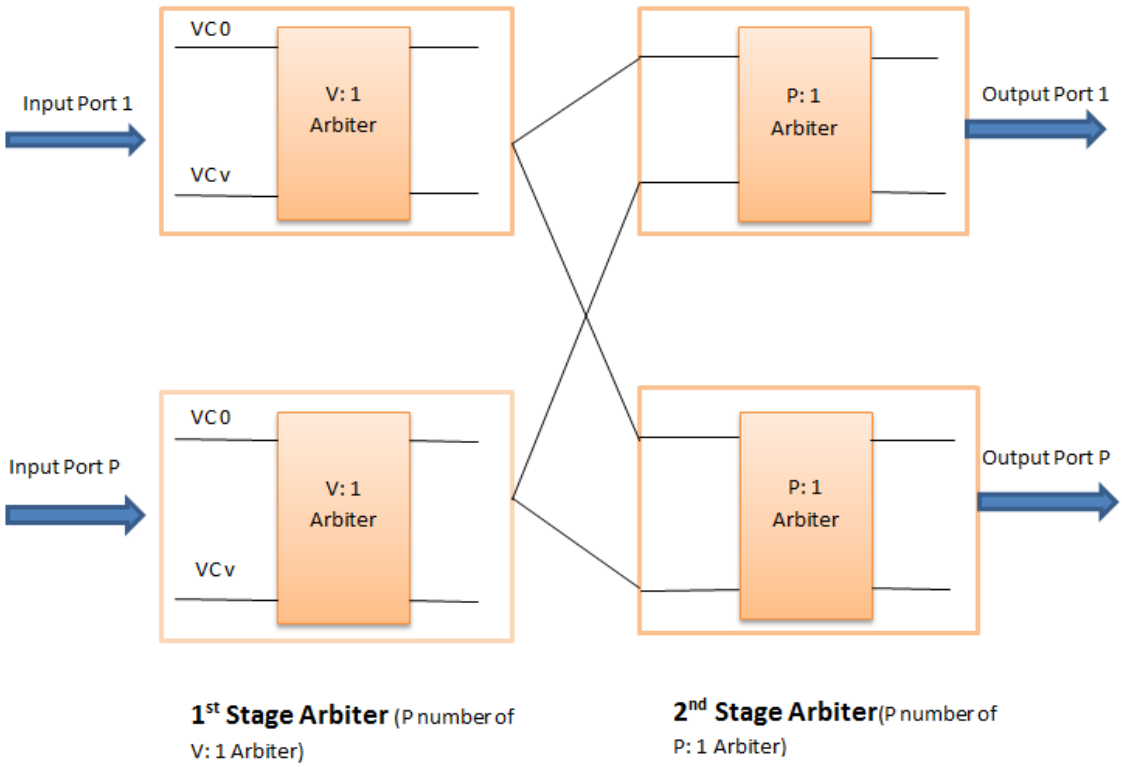

Figure 3.4: Basic switch arbitration

Figure 3.4 represents the switch arbitration in baseline NoC. It consists of two stages. In the first stage, one VC is selected among v number of VCs of each input port. It needs $p(v:1)$ arbiters if there are p number of input ports. In the second stage, it selects one input port among p input ports for each put ports, considering there are p number of output ports. So, total number of arbiters in second stage is $p(p:1)$.

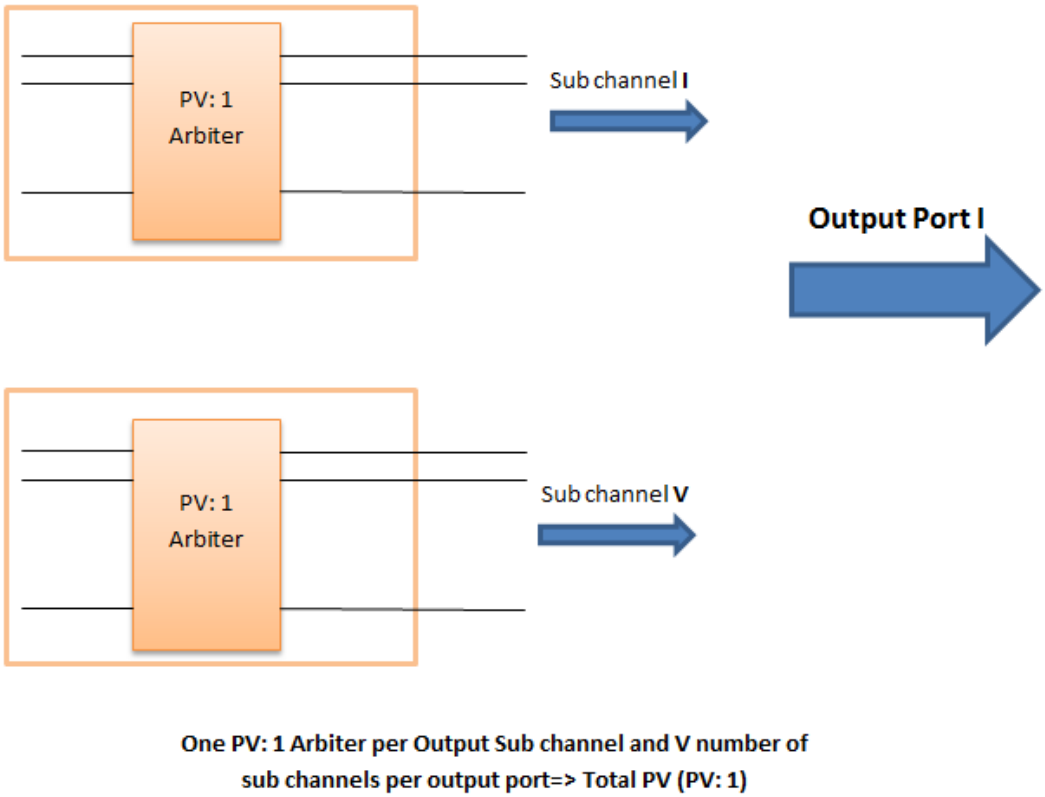

Figure 3.5: Overview of switch arbitration. Reprinted with permission from [5]

Figure 3.5 shows the switch arbitration of channel-stealing scheme which has the most complex SA structure among all APCR schemes. It is achieved by single stage arbitration and the virtual channels does not own any dedicated sub-channels, which

means one VC can access other VC's sub-channel in same output port if that sub channel is not assigned to its corresponding VC from any input ports. Hence, if p be the total number of input ports and v be the total number of VCs, then the output arbiter maps one VC from pv number of requests for one articular output subchannel.

They adopted round-robin based scheduling algorithm to allocate physical sub-channels among different requesting flits from each individual input buffer that has request to the same output direction which is not an optimal scheduling policy in terms of quality of service requirement.

3.3 Switch Arbitration and Scheduling Policy in APCR

Switch arbiter is the fundamental component in NoC routers which comprises of a number of shared resources in the whole process such as switch ports, sub channels, and physical bandwidth etc. Scheduling is performed at each stage of switch arbitration starting from choosing appropriate of VC of each input port to choosing appropriate input ports for each output arbitrator. Scheduling can be static or dynamic depending upon performance requirement. In Static-Priority scheduler, the priority is constant throughout the process and never changes at run time. So priority is fixed before the selection process starts, where as, dynamic scheduler determines the assignment and ordering of tasks at run time. Generally, dynamic priority leads to a better solution, but computational overhead in selection process and switching of control from one candidate to other will increase delay and the energy consumption of network dynamically. As a result, static scheduling is recommended for NoC. There are still good quality of scheduling algorithms for NoC which is still a topic of research today. At the same time scheduling algorithms may be preemptive or non-preemptive depending upon dynamic and static priorities. That means preemption of control

from one VC to another VC in order to send the flits relays upon the scheduling policy. Scheduling problem is an NP hard problems in NoC. These problems are bounded by constraints like latency delay, fairness and throughput etc as per the service requirements. However, schedulers like round-robin is best known for providing good fairness to the candidates.

Now, we take a look at previous scheme, round-robin which fails to provide quality of service. For instance, round-robin scheduler considers providing fairness by cyclically traversing among different VCs at all input port. It maximizes the overall system throughput by assigning resources cyclically to the users without considering current network status. One of the cases where the round-robin fails is shown in figurative way below.

Figure 3.6 represents a situation where round-robin fails to be an optimal scheduling policy. As the figure shows, we can see VCs from three input ports are requesting the same output port as shown. First VC from IP_1 has four flits to send where as others have maximum one or two flits of same packet in current cycle. So, in round-robin, the requests are processed in cyclic manner. When one VC from IP_1 will have three flits after sending in current cycle, in next cycle, it will take flits from next VC from IP_2. If the packets in IP_1 have critical flits to send, then round-robin is most likely to miss their deadlines. Again, in next cycle, the empty VCs may receive more packets which will again delay the traversal of critical flits.

Therefore, round-robin policy does not provide best service, because in next cycle, when there are more flits left in VC of $IP1$, it tries to forward packets from other input ports that have actually less flits left because of their turn priority. In mean time flits may arrive in between VCs and are selected for the arbitration. All these scenarios lead to delay in the selection of the very first VC and the latency of the flits in that VC increases as each cycle leads. This is a very simple procedure which gives

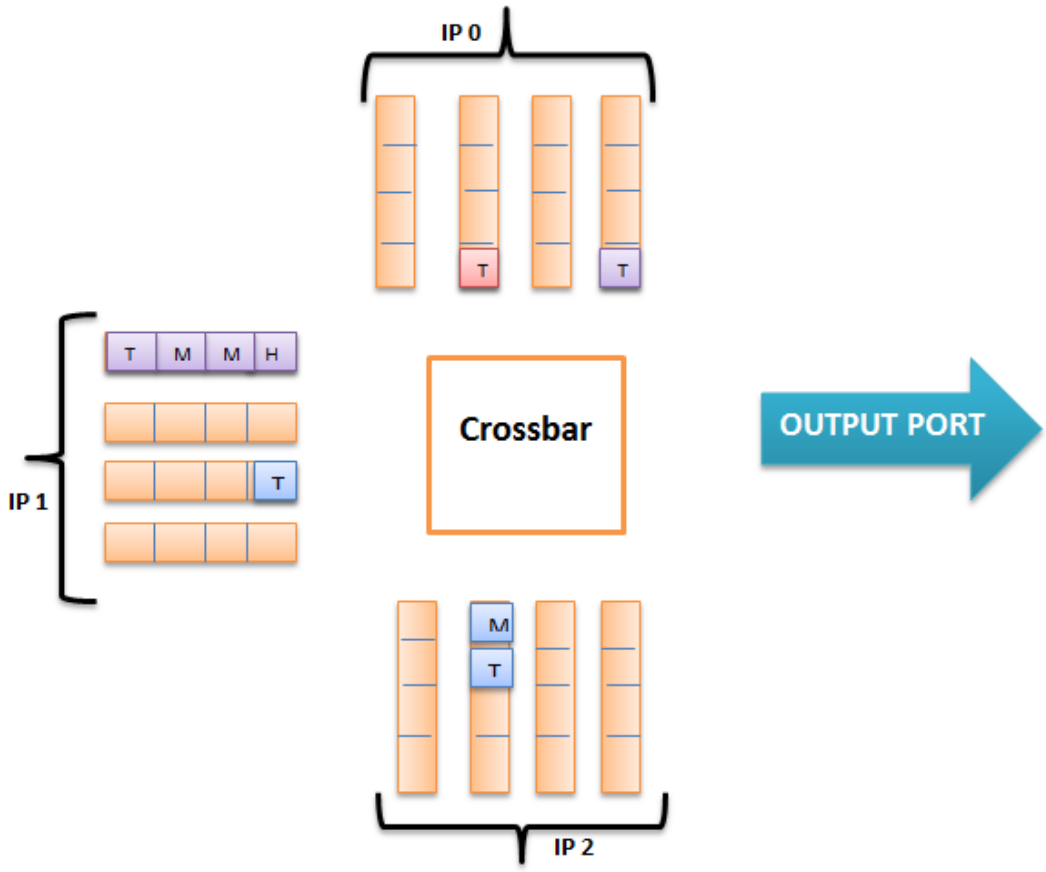

Figure 3.6: A case where round-robin fails to provide good quality of service

best fairness to the system, but it exploits poor performance in terms of QoS. This is the motivation of our work. We try to customize the scheduling policy in switching arbitration of channel-stealing so that it will provide fair chance to all input port and thus each VCs so as to provide better *QoS*. Generally, *QoS* is being ensured by satisfying a number of performance metrics like latency, area, throughput, fairness when the link bandwidth is distributed among various cores of NoC. In our work, we try to highlight the quality of service requirement and minimizing latency delay along with high throughput in the respected environment. In next section, we discuss

about our proposed scheduling policies where we explain, demonstrate, and prove the correctness of each techniques.

4. RELATED WORK

Our work is based upon APCR [5] that provides a novel switch allocator policy for NoC. It is observed that the channel stealing scheme of APCR gives the best performance among all the schemes, such as fair sharing, monopolizing. In channel stealing, the dedicated bandwidth of one particular VC can be utilized by other VC if the owner of the sub channel does not have any flits to send. They used round-robin scheduling policy to schedule flits at crossbar port. There are other related implementation to provide best services to incoming packet requests. These techniques deals with wither resizing the network in order to satisfy all system requirements by providing enough bandwidth to the network, or impose priority based scheduling at SA or VA stage in router pipelining to get a good matching. Guerrier and Greiner [6] proposed a general architecture for system-on-chip. Daniel and Dally [7] analyzed different existing architectures, based on varying parameters. He also proposed two schemes; one on speculative switch allocation scheme and other on VC allocation. Mukherjee et al. [8] compared the three schemes for switch allocation in interconnection networks. They also proposed a priority rule which helped the network not to saturate at high workloads. Keslassy et al. [9] has designed a new scheduler for low jitter requirements. Raina and Muthukumar [10] implemented scheduling, based on the traffic between the nodes at all times. Ji et al. [11] designed a small switch with an aim to minimize the system utilization. They prove that max-weight policy is not optimal for high traffic. Some of these proposed schemes include Andreasson and Kumar [12] and Kim et al. [13] that propose a slack-time aware routing to improve overall system utilization that dynamically controls the packet injections in the networks to achieve QoS. Again, in contrast to existing round-robin schemes,

there are also advanced version of round-robin that has been proposed. For example, an optimal round-robin scheme in arbiter design is implemented by Jou and Lee [14]. Towles and Dally [15] presented three techniques that would provide performance guarantees for scheduling switches with configuration overhead. Winter and Fettweis [16] introduced a global communication resource allocator working on task level. It reserves VCs dynamically at run time between two sub modules through out the NoC providing theoretical latency and bandwidth. Guderian et al. [17] proposed Age based scheduling and weighted round-robin technique to provide fairness in bandwidth allocation. Abts and Weisser [18] proposed an Age based algorithm for providing fairness among the nodes. Similarly, various arbitration policies as in Yum et al. [19] and Chien and Kim [20] have been proposed in chip multiprocessors to provide fairness while we try to re implement our arbitrator that can satisfy both real time and non real time performance with guaranteed quality of service without degrading fairness and overall system throughput. Again, fairness has been intended in several switch arbitration design. "Approximated Age based packet arbitration" proposed by Lee et al. [21] satisfies QoS, but it has does not provide enough fairness at bandwidth allocation. This motivates us to introduce new priority based switch arbitration schemes which not only satisfies the QoS, but also provide the fairness. McKeown [22] proposed the arbitration scheme which is based on a maximal size approach and a variant of round-robin matching to prevent starvation under uniform traffic. Balakrishnan and Ozguner [23] proposed priority based arbitration methods. None of these scheduling policies in Network on chip paradigm could provide fairness and QoS at same time in NoC. So, we try to implement dynamic priority based scheduling policy which attempts to meet both the requirement at same time.

5. OUR PROPOSED SCHEDULING POLICY

Implementing an efficient scheduling algorithm requires a variety of factors in NoC to be considered, like the type of application, the power, energy and area budget and QoS requirement etc. Depending on the system environment, we might expect the scheduler to maximize the throughput of network, which means it should be capable of serving maximum number of candidates per each cycle. It should avoid indefinite blocking or starvation among packets in virtual channels. It should minimize overhead and be able to enforce priorities. QoS involves fair allocation of resources under some service policy where the fairness may be further categorized as latency fairness or throughput fairness. The existing round-robin policy was able to fairly allocate the resources in order to maximize the throughput where each VC gets a fair chance to utilize the resources. But, this policy is not able to satisfy the latency fairness requirement where each packet needs to have guaranteed services to be processed in time. So, it is not able to handle the critical packets to meet their deadlines on time. Also, supporting the diverse application in on chip networks is another motivation behind our work. So, we try to implement three scheduling policies that satisfy diverse requirements in APCR environment, such as proportional fair algorithm (PF), Age based Scheduling policy, and Static-Priority based scheduling policy (PR). In next section, we present our switch allocation policy which is same for all scheduling policies.

Our algorithm is performed in four vital steps as demonstrated in figure 5.1. In first step, for each input port, we store the requests from the VCs in a vector $vec1 < VC\#, Req >$. In second step, we search for any empty cell in the vector $vec1 < VC\#, Req >$ obtained in previous step in order to check if any of the

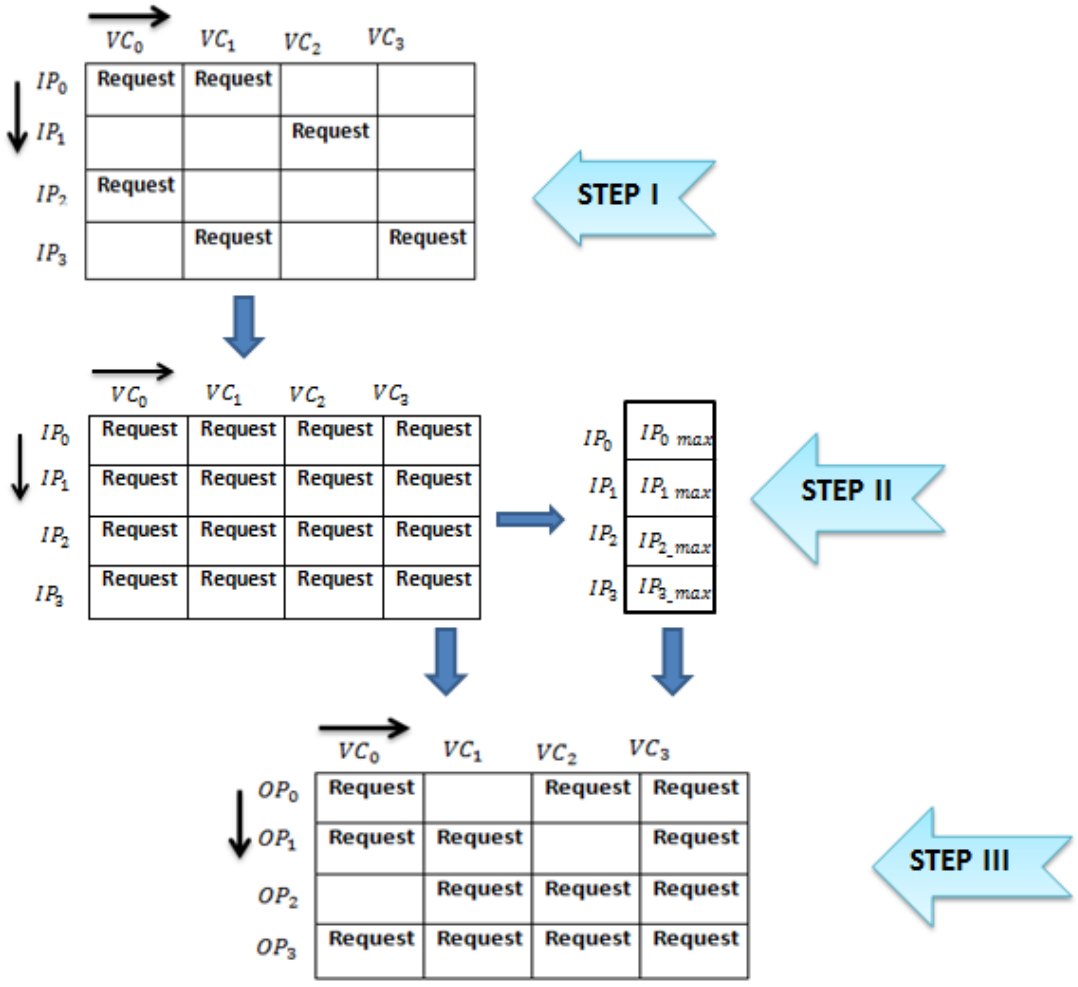

Figure 5.1: Switch allocation flow diagram

VCs from that input port has no request to send in the current cycle. We apply our proposed scheduling technique in order to fill up the unfilled entry in $vec1 < VC\#, Req >$ cells. This is possible due to channel-stealing scheme which allows other VC or input buffers to utilize the sub channels that they don't own. In next step, we have the vector $vec1 < VC\#, Req >$ for each input port filled with all the valid requests from input ports. Now, we find out the maximum priority among all VCs for the particular input port depending upon the scheduling policy and

the credit in selected VC represents the input port as IP_{max}. The types of credit depends upon the types of scheduling policy we choose which will be discussed in next section. So we generate a vector that has the list of all IP_{max}. In prior to last step, we have the vector $vec1 < IP, VC >$ that stores all the considered requests for switch allocations. The structure of these vectors at each step is figuratively shown below. In last step, we generate a resultant vector $vec3 < outport, VC >$ that keeps the final granted request from all input port VCs. It starts with each input port and determines the IP_{max} from each input port that has request to the same output port. If another IP_{max} of different input port requests the same vector entry, we again apply our proposed scheduling algorithm as we do in second step. Finally the $vec1 < VC\#, Req >$ is filled up with all valid requests in final step, before putting the requests in vector $vec1 < VC\#, Req >$.

5.1 Proportional Fair Scheduling

5.1.1 Prior Study

Our first scheduling policy is known as proportional fair scheduling which allocates more resources to the requesting candidate that has deserving network status for that cycle. hence this is a dynamic priority based scheduling policy which is based on two dynamically changing factors of each candidate input port. One of them is the number of flits that are present in them as that have direct impact on matching decision. Secondly, the number of flits that the input port/VC has already processed by previous cycle. We calculate the number of flits that is currently in switch arbitration stage. For each input port, we traverse all four VCs. The algorithm is broadly discussed in next section.

5.1.2 Proposed Algorithm

We calculate the ratio for each VC in each cycle depending upon the network status. The ratio can be calculated as the number of flits that are currently requesting for the mapping divided by the average throughput of the VC. The average throughput of each VC is nothing, but the average number of flits that are already been sent till last cycle. After finding the ratio of each VC for that input port, we sort them in increasing order of respective ratios. The VC with highest ratio gets highest priority for that current cycle. This is the first stage of dynamic priority calculation. Second stage of calculation is calculating ratio for each input port. Total number of flits currently requesting in an input port is the sum of requests from all VCs belonging to the same input port. Similarly, the total number of flits that are already been sent from the current input port is the sum of flits sent from all its VC. Then we calculate ratio for each input port in similar way which is the total flits requesting from the input ports for that particular cycle divided by average throughput of the input port considering all VCs from same input port. We sort the input port in increasing order of their respective ratio. When switch arbitration starts for one output port, it looks at the highest ratio input port and gives the priority. In the selected input port, we choose the highest ratio VC. We have taken phit size to be 4 times of one flit size. If it does not find 4 flits from the selected VC it jumps to next highest ratio VC and so on. After traversing all VC from the same input port, if it still does not get four flits, it jumps to next highest ratio input port and so on.

5.2 Age Based Scheduling Policy

5.2.1 Prior Study

We now discuss our second scheduling policy. The motivation behind our scheduling algorithms is again to provide best service to packets among different buffers.

Unlike PF scheduler, it does not have to store the record of large number of flits in each VC. It otherwise allocates the flits, based on their time of injection into the network. It is an attempt to reduce the overall packet latency of each packet in the network. Now, we present the algorithm and give a clear demonstration.

5.2.2 Proposed Algorithm

If a packet has been has arrived before another packet arrives, then overall latency of the former packer will be more than the packet arrived later. So as to be fair in the process, we consider the time of injection of all the packets (flits) into the network. For example, there are two VCs requesting same sub channel. First, this algorithm finds out the time of injection of the candidate packet. Here the injection time of each packet is same as the injection time of all its corresponding flits. So the age is defined as the difference between the time of injection and the current simulation time. The algorithm gives the highest priority to the packets that has the highest age. Because, delay in selecting the packet in arbitration leads to increasing age of the packet and thus the latency and criticality of the packet increases.

5.3 Static-Priority Based

5.3.1 Prior Study

We also investigated performance of APCR in channel-stealing imposing static-priority to each incoming packet at its injection time. The concepts revolves around providing best services to real time traffics where the network needs to ensure the packet deadline while trying to minimize each packet latency delay. In this regard, there are various priority based arbitration methods have been proposed as in [23]. In our case, we consider flit level priority based preemption method.

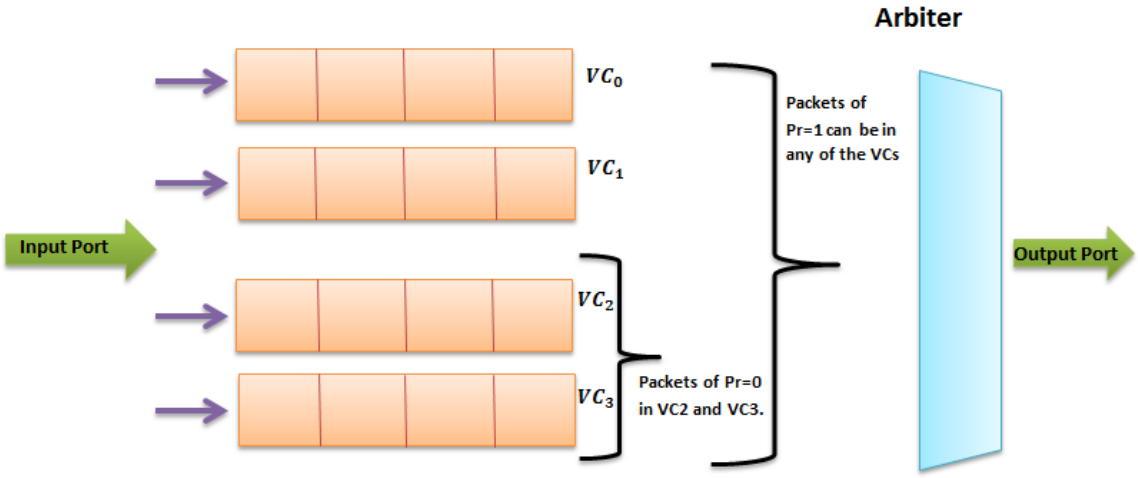

Figure 5.2: Static-Priority based virtual channel allocation

5.3.2 Proposed Algorithm

In our implementation, we assign one Static-Priority value to each packets which can be zero or one, in our case, where zero is the low priority and one is the high priority value. This technique is also been implemented at virtual allocation stage before SA stage unlike other two schedulers we discussed before. Here we assign different priority to each VC depending upon the priority of packet residing in them. There are four virtual channels in our algorithm like as in APCR.

The figure shows the impact of Static-Priority based scheduler in arbitration process. As we mentioned, there are four VCs and when packets are injected into the network, we randomly prioritize the packets. The packets can have a high priority (Pr = 1) or a low priority (Pr = 0). So, the main idea behind this policy is to limit the accessibility on output VCs of current packets depending upon the priority. High priority can be forwarded to any of the output VCs where as low priority packets can be forwarded to limited number of output VCs.

For example, the high priority packets can access all VCs starting from VC_o to VC_3 where as the low priority packets can only access two VCs, such as VC_2 and VC_3. In synthetic traffic, we have assigned priority randomly as they arrive. So, this can be applied to real time traffic where we can assign critical or delay bound packets to bear high priority and non critical packets can be assigned low priority. The Static-Priority based scheduling is initiated as the packets get injected. The process starts with assigning the priority to the packets at injection time. Then unlike other schedulers that are previously being explained, we also deal with virtual channel allocation stage. So, when the switch arbitration starts, it first look at the virtual channels such as VC_o and VC_1 and if there are no packets to send at that time, then it looks for VC_2 and VC_3 if there are packets to be sent. This order is because of the fact that the high priority packets can be in any of the four VCs where as VC_o and VC_1 are guaranteed to have high priority task and VC_2 and VC_3 will have either high or low priority task. So to give guaranteed service to the higher priority task, we start with traversing VC_o and VC_1.

Algorithm 1 Switch allocation

initialize i, j, where i is the input index and j is the VC index.
initialize vector $vec1 < IP_i, VC_j >$that will keep the requests coming from virtual channel $VC_j$of input port IP_i.
1. for each input port IP_i, **do**
for each $VC_j$of selected input port, **do**
if VC_j has request, **do**
store Request from VC_jin $vec1 < IP_i, VC_j >$
end if
end for
end for
2. for each input port IP_i, **do**
if entry in $vec1 < IP_i, VC_j >$is zero, **do**
find requests from other VC of same IP_i
find best VC among all requests using scheduling algo and fill the $vec1 < IP_i, VC_j >$
end if
end for
3. initialize vector $vec2 < IP >$
for each input port IP_i, **do**
find $(VC_{max})_i = max\{vec1 < IP_i >\}$
for (IP_i, VC_j), do
$IP_{imax} = IP_i$
store IP_{imax} in $vec2 < IP_i >$
end for
end for
4. initialize vector $vec3 < OP_k, VC_l >$, where $k > 0$ and $l > 0$
for each output port OP_k, **do**
for each entry $(IP_i VC_j)$ in $vec1 < IP_i, VC_j >$,**do**
find corresponding output port $OP_k$and VC_l,
for each $OP_k$and VC_l in $vec3 < OP_k, VC_l >$, **do**
if there is no entry, **do**
put the corresponding request Req of $(IP_i VC_j)$ in requested entry in$vec3 < OP_k, VC_l >$
else if there is already an entry, **do**
call scheduler and select any one request among the two.
end if
end if
end for
end for
end for
return the resultant vector $vec3 < OP_k, VC_l >$

Algorithm 2 PF-Scheduling Algo

for each input port IP_i, **do**
  for each virtual channel VC_j, **do**
    find $average\,throughput$ of the VC = Number of flits that are already sent through the VC
    find $flits_{curr}$= Total number of flits belonging same packet
    Calculate Ratio R_{ij}= $\frac{flits_{curr}}{average\,throughput}$
  end for
  sort the $R_{ij}$and keep in a vector $vec < VC\#,\ R_{ij} >$
end for
for each input port IP_i, **do**
  find $average\,throughput$ of the input port IP_i = Number of flits that are already sent through it
  find $flits_{curr}$= Total number of flits belonging same packet that has requests.
  Calculate Ratio R_i= $\frac{flits_{curr}}{average\,throughput}$
end for
sort the $R_i$from all input ports and keep in a vector $vec < IP\#,\ R_i >$
Return sorted $vec < IP\#,\ R_i >$

Algorithm 3 Age based scheduling

for each input port IP_i, **do**
  for each virtual channel VC_j, **do**
    find the request from the VC and find the injection time of the flit i.e. $T_{injection}$
    find $age = T_{current} - T_{injection}$
  end for
  sort the age and keep in a vector $vec < VC\#,\ age >$
end for
for each input port IP_i, **do**
  find the maximum age $age_{max} = max\{vec < VC\#,\ age >\}$
end for
Find the age_{max} for each input port and keep in a vector $vec < IP\#,\ age_{max} >$
sort the $vec < IP\#,\ age_{max} >$ from all input ports
Return sorted $vec < IP\#,\ age_{max} >$

Algorithm 4 Static-Priority based Scheduling Algorithm

Static-Priority based Algorithm at VA Stage:
for each packet P, **do**
    Find the priority of packet at Head Flit
    if the priority of the packet is 1, **do**
      for $j := 0\, to\, 3$, **do**
        if VC_j has credit = 1, **do**
            assign $VC_j$to the packet p.
          Break
        end if
      end for
    else if the priority of the packet is 0, **do**
      for $j := 2\, to\, 3$, **do**
        if VC_j has credit = 0, **do**
          assign $VC_j$to the packet p.
          break
        end if
      end for
    end if
end for
Static-Priority algorithm at SA Stage:
for each input port IP_i, **do**
    for $j := 0\, to\, 1$ in virtual channel VC_j, **do**
      if packets present in the VC, **do**
          select the packet for switch allocation
      end if
    end for
    for $j := 2\, to\, 3$ in virtual channel VC_j, **do**
        search for the VCs that have packet of priority 1.
        if packet of priority 1 is found, **do**
          select the packet for switch allocation
          else if packet of priority 1 is not found, **do**
              select the VCs that have packet of priority 0.
          end if
        end if
    end for
end for

6. SIMULATION RESULTS

In the section, we explain a comprehensive evaluation of our three proposed scheduling policies. First we present the evaluation standard of our simulation process in Section 5.1. We have evaluated the new algorithm considering two important performance metrics, First, QoS, which is defined in Section 5.2.1 and second, fairness of system in Section 5.2.2. We also compared our all three proposed scheduling policies with round-robin scheme with simulation results in Section 5.

6.1 Evaluation Methodology

We have used a cycle accurate NoC simulator to measure the packet delay for all our proposed scheduling policies and compared their average packet latencies with round-robin scheme. We use an 8x8 network where the buffer size is in the order of number of flits for each VC is set. The links are modeled as 512 parallel wires. The APCR router uses a 128-bit flit. In simulation, there are two kinds of packets in the network, such as control packets (short packets) and data packets (long packets). Each short packet consists of one-flit and a long packet consists of five-flits. We have used XY routing algorithm that is appropriate for our scheduling schemes, and configured four VCs for each input port. The CMP system parameters such as clock frequency are set to 4GHz, $L1$ I and D cache used are 1-way and 4-way with each 32KB, 1 cycle. $L1$ and $L2$ cache are of 64B. Memory latency is 300 cycles. Then $L2$ cache is taken as 16 ways, 16MB and each bank is 512KB where there are 32 banks. We have considered synthetic traffics for our evaluation such as Transpose (TP), Uniform Random (UR), Bit Complement (BC) in which the content of short packets is 60%. Our CMP configuration with static non uniform cache architecture [13] has 32 out-of-order processors and 32 $L2$ cache banks in a

single chip. On-chip interconnection network with 2D Mesh topology connects all processing elements (cores) and their associated $L2$ cache banks. Again, CACTI [24] measures all cache delays and area parameters. The table shown below presents basic network configuration and CMP parameters taken in our simulation.

Table 6.1: Basic Network Configuration

Characteristic	**Value**
Topology	8x8 2D Mesh
Routing	XY Routing
Router Architecture	APCR
Per hop Latency	3 cycles: 2 cycle in a router, 1 cycle to cross a link
VC/Ports	4
Phit Size(Channel-width)	512 bits
Flit Size	128 bits
Packet Length (flits)	1 (control packet), 5 (data packet)
Traffic Pattern	Uniform Random, Bit Complement, Transpose
Simulation Warm-up Cycles	10,000
Total Simulation Cycles	200,000

Table 6.2: CMP System Parameter

Clock Frequency	4GHz
L1 I & D Caches	1-way & 4-way, 32KB, 1 cycle
L2 Cache	16-way, 16MB, 512 KB/bank, 32 banks, 20 cycles
Memory Latency	300 cycles

6.2 Simulation Results

We have evaluated the average packet latencies under three traffic patterns such as bit complement, transpose, and uniform random. Figure 6.1 till figure 6.3 show the simulation results of baseline, APCR router and all three scheduler for channel-stealing under bit complement workload. Then figure 6.4 till figure 6.6 show the simulation results of baseline, APCR router and all three scheduler for channel-stealing under TP workload. Again, figure 6.7 till figure 6.9 show the simulation results of baseline, APCR router and all three scheduler for channel-stealing under UR workload. Figure 6.10, figure 6.11, and figure 6.12 give a comparison view of all four scheduling algorithms i.e. round-robin, PF-based, Age based, and Static-Priority based scheduler using all three traffic pattern such as BC, TP and UR. The results are consistent with our expectations. We have now considered different applications as in BC, TP and UR with different priority notions. In previous implementation in [5], they assume single type of applications where every packet has same priority. In out implementation, we have considered two types of prioritized applications in which PF and Age based scheduler provides single priority type application service. Then, Static-Priority scheduler that randomly imposes random priority to incoming packets. It is shown that, as we increase the workload, the performance gets better in all scheduler as compared to round-robin scheme although there is not much difference in performance between PF and Age based scheduler. So, our scheduling policies work very well at high workload where we can see PF and Age based scheduling policies seem to be promising under BC and UR workload. When the packet injection rate is low, the performance of the four scheduling policies only has minor differences. Our scheduling policies work very well at high workload. As the chances of competition among candidates is very low, so round-robin policy is fair

enough for switch arbitration at low load condition. Below, we have explained the performance improvement for each scheduler under all possible workloads for the above three traffic patterns.

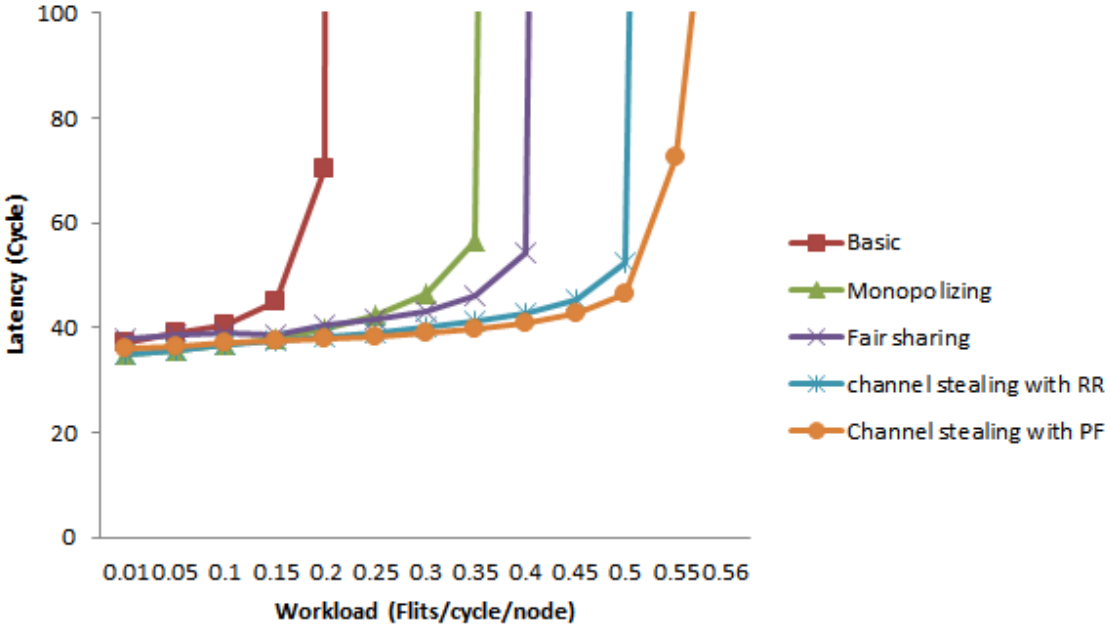

Figure 6.1: Performance of PF scheduler under BC workload

Figure 6.1 shows the simulation results for PF-based scheduling using BC traffic from 0.01 (very low load) to 0.56 (high load) injection rates. The result is consistent with our expectation. It is seen that, for BC traffic pattern, there is 12% performance improvement from round-robin in case of PF scheduler where as the PF scheduler outperforms the baseline router by 28x performance improvement.

Figure 6.2 shows the simulation results for Age based scheduler using bit complement traffic from 0.01 (very low load) to 0.56 (high load) injection rates. The result is consistent with our expectation. It can be seen that, when the packet injection

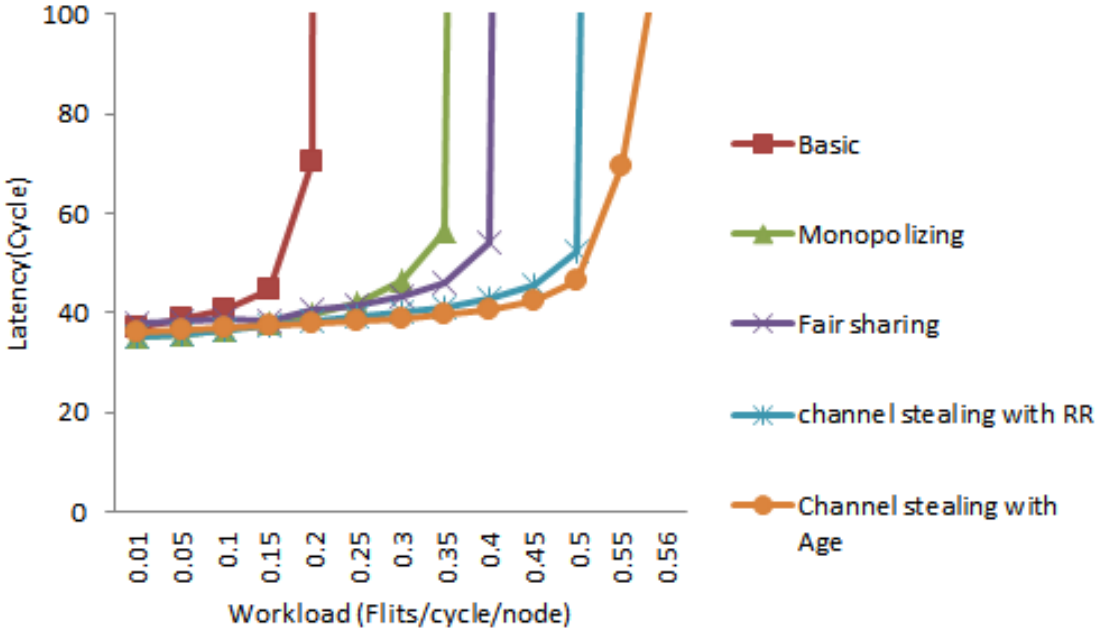

Figure 6.2: Performance of Age based scheduler under BC workload

rate is high, the performance of Age based scheduler outperforms the round-robin scheme and as well as fair-sharing, monopolizing and baseline router. We can see, among all schemes, channel-stealing is the best under Age based scheduler for this particular traffic criteria as shown in the above figure. It is seen that, for bit complement traffic pattern, there is 12% performance improvement from round-robin in case of PF scheduler where as the Age based scheduler also outperforms the baseline router by 28x.

Figure 6.3 shows the simulation results for priority based scheduling using bit complement traffic from 0.01 (very low load) to 0.55 (high load) injection rates. It can be seen that, when the packet injection rate is high, the performance of Age based scheduler is same as round-robin scheme. We can see, round-robin works better than Static-Priority based scheduler at both high workload. However, Priority

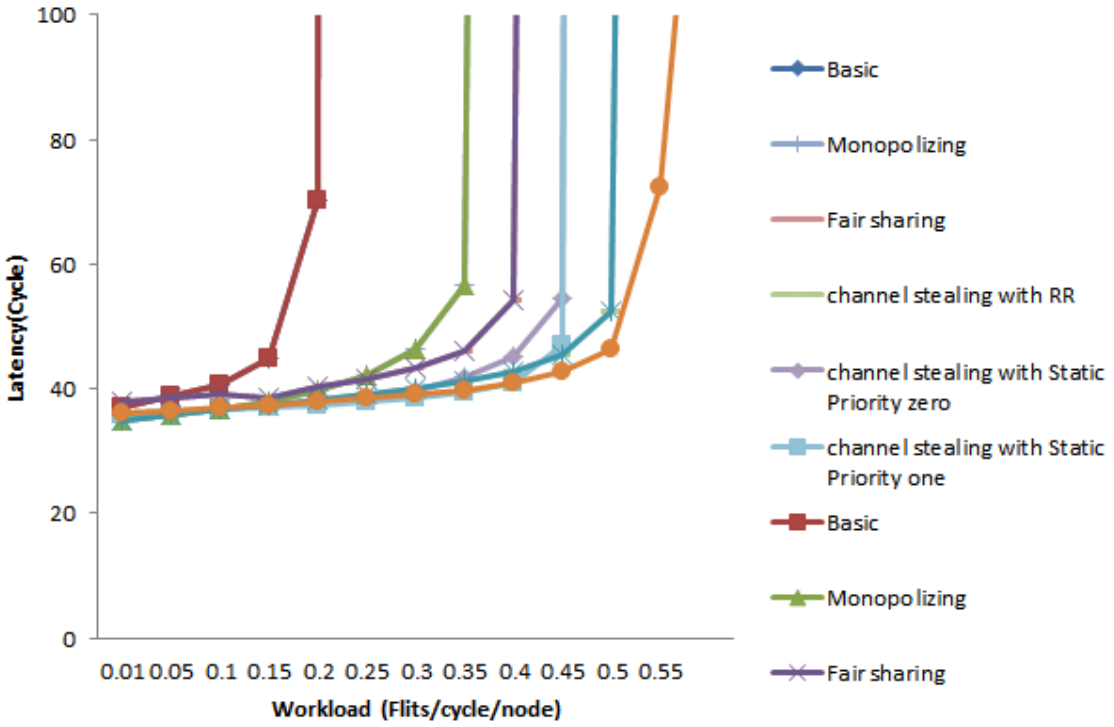

Figure 6.3: Performance of Priority based scheduler under BC workload.

based scheduling works better at middle injection rate such as 0.2 to 0.4 for this particular traffic criteria as shown in the above figure. Again, the Static-Priority based scheduler outperforms the baseline router by 3X performance improvement.

Figure 6.4 shows the simulation results for PF-based scheduling using transpose traffic from 0.01 (very low load) to 0.56 (high load) injection rates. The result is consistent throughput all the workloads in the sense that the round-robin and PF based scheduling for channel-stealing in APCR performs almost similar. Both of them works better than any other APCR or baseline routers for this particular traffic scenario.

Figure 6.5 shows the simulation results for Age based scheduler using transpose traffic from 0.01 (very low load) to 0.56 (high load) injection rates. The result is

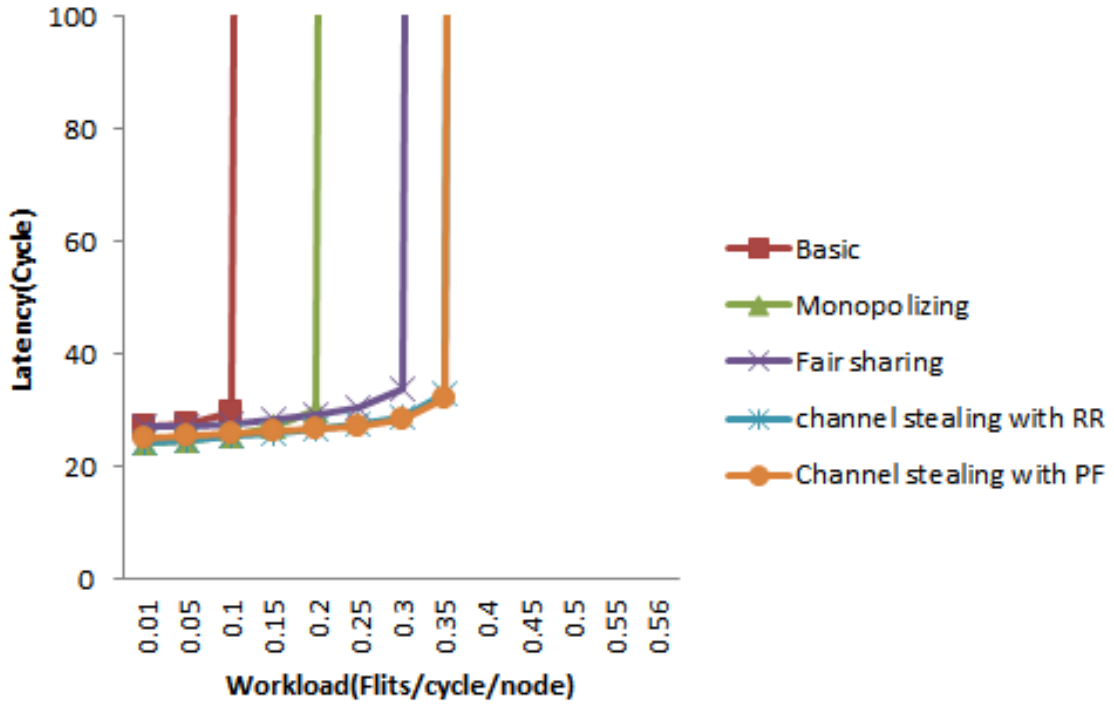

Figure 6.4: Performance of PF scheduler under TP workload

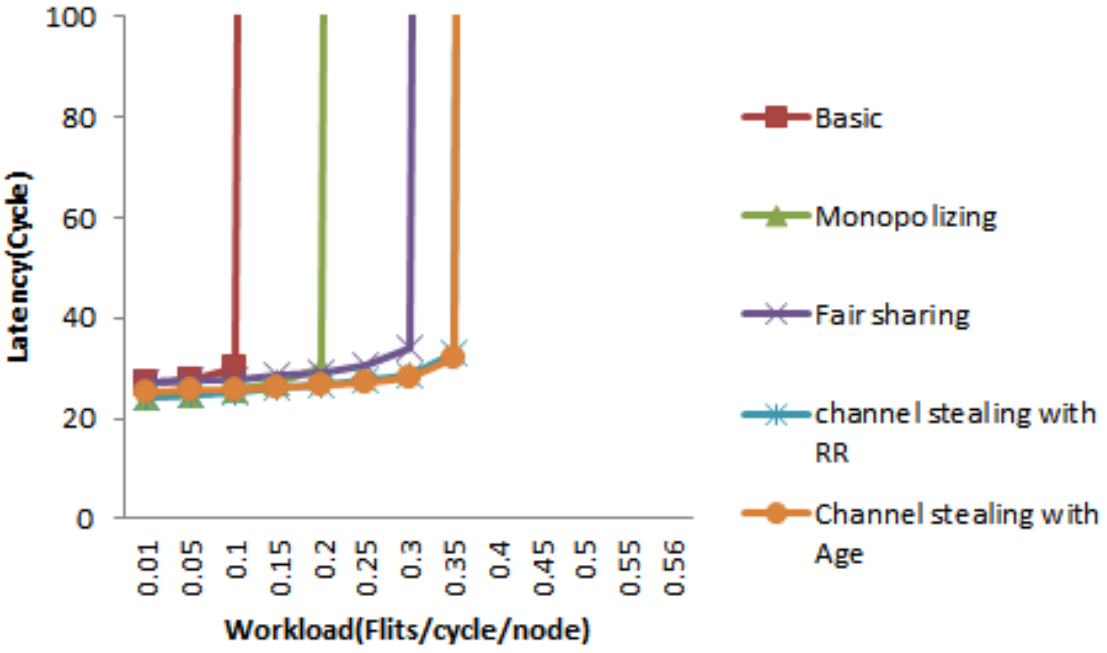

Figure 6.5: Performance of Age based scheduler under TP workload

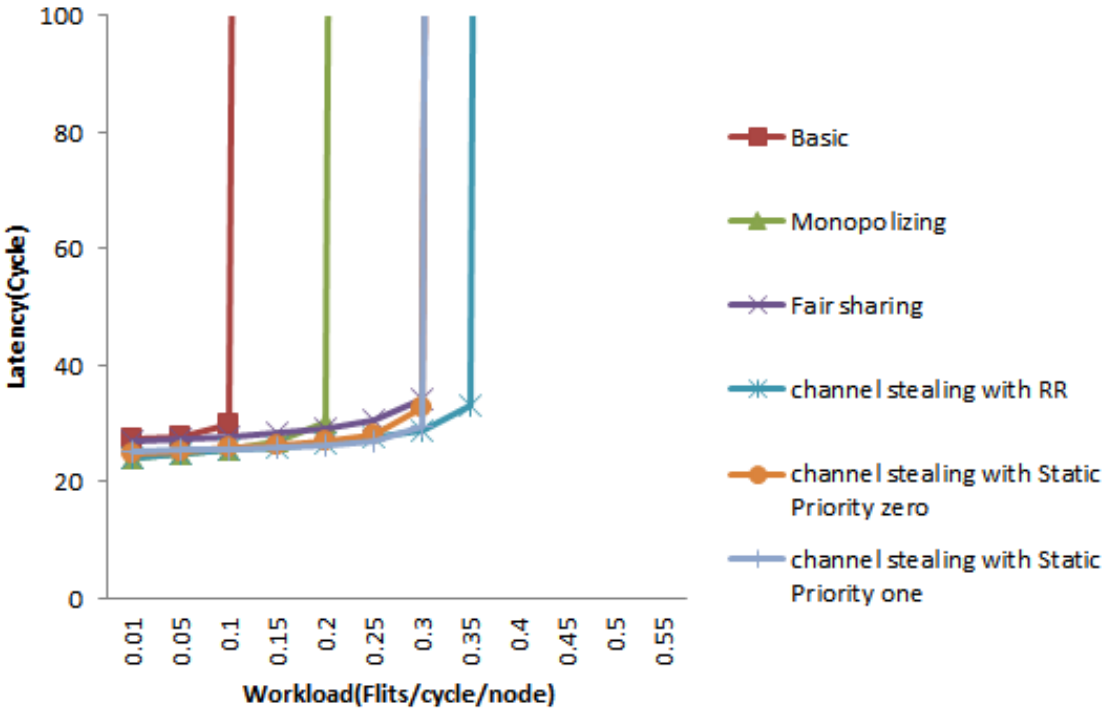

Figure 6.6: Performance of Priority based scheduler under TP workload

consistent throughput all the workloads in the sense that the round-robin and Age based scheduler for channel-stealing in APCR performs also very similar as in PF scheduler. Both of them works better than any other APCR regulation scheme besides channel-stealing or the baseline routers for this particular traffic scenario.

Figure 6.6 shows the simulation results for Priority-based scheduling using transpose traffic pattern from 0.01 (very low load) to 0.55 (high load) injection rates. Surprisingly, round-robin performs better than priority based scheduler in both high and low priority cases. However, round-robin works better than priority based scheduling and any other APCR or baseline routers for this particular traffic scenario.

Figure 6.7 shows the simulation results for PF-based scheduling using uniform random traffic from 0.01(very low load) to 0.8 (high load) injection rates. The result

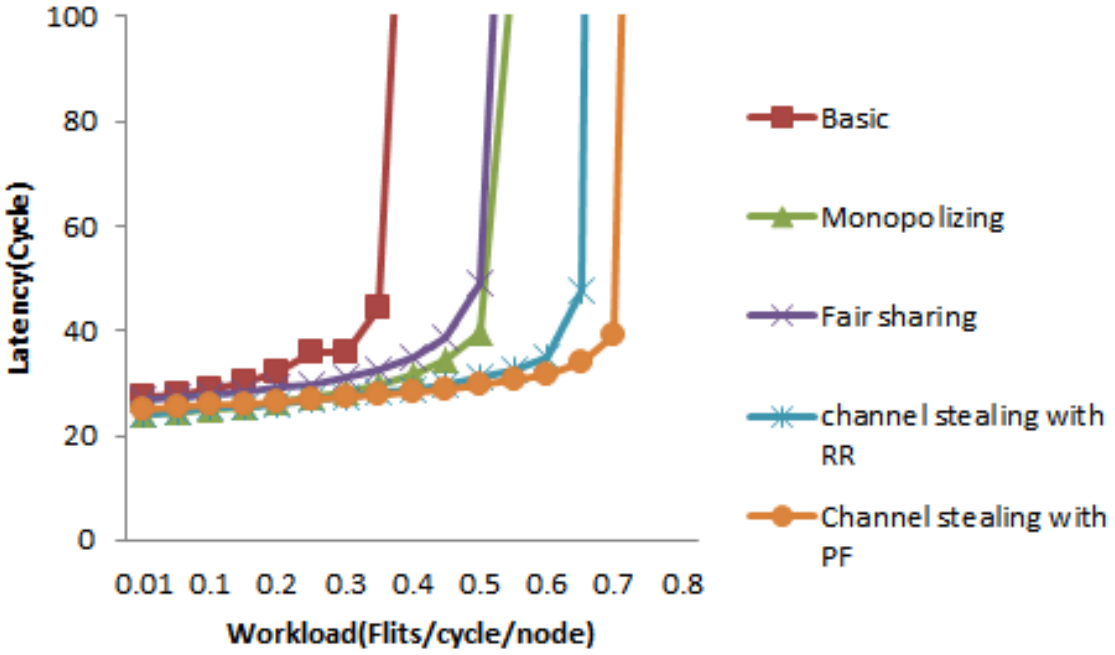

Figure 6.7: Performance of PF scheduler under UR workload

is consistent with our expectation. It can be seen that, when the packet injection rate is high, the performance of proportional fair (PF) scheduler outperforms the round-robin scheme and as well as fair-sharing, monopolizing and baseline router. We can see, among all schemes, channel-stealing is the best under PF scheduling for this particular traffic criteria as shown in the above figure. It outperforms the baseline router by 2X performance improvement. Also, compared to round-robin scheme there is 15% performance improvement in PF scheduler under UR traffic.

Figure 6.8 shows the simulation results for Age based scheduler using uniform random traffic from 0.01 (very low load) to 0.8 (high load) injection rates. The result is consistent with our expectation. It can be seen that, when the packet injection rate is high, the performance of Age based scheduler outperforms the round-robin scheme and as well as fair-sharing, monopolizing and baseline router. We can see, among all

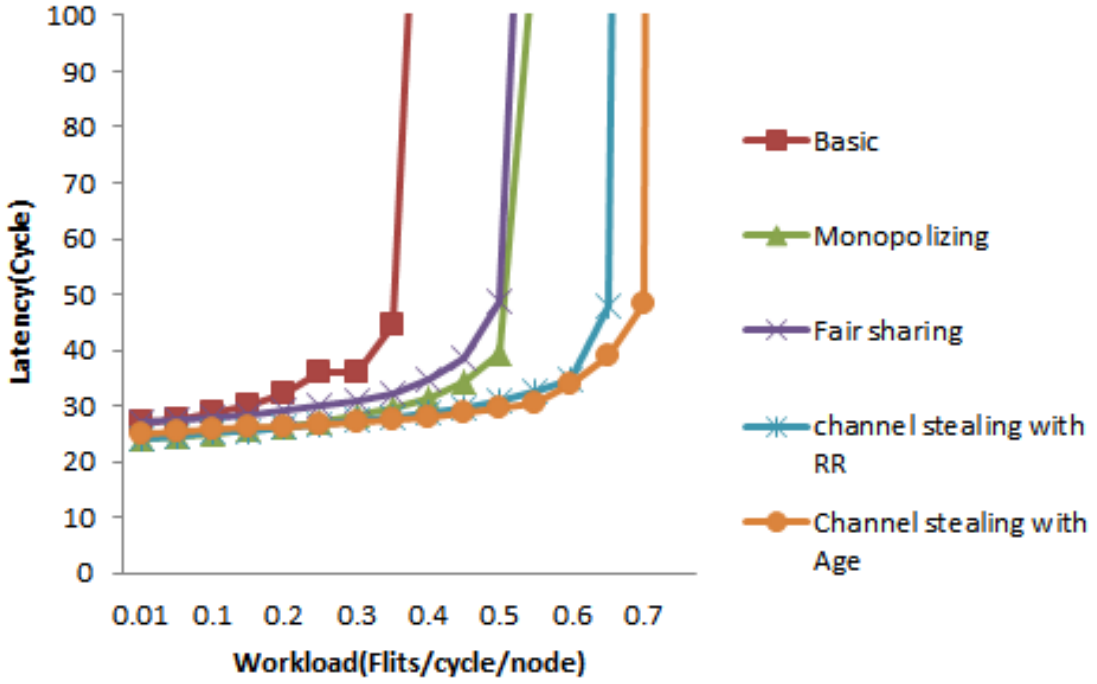

Figure 6.8: Performance of Age based scheduler under UR workload

schemes, channel-stealing is the best under Age based scheduler for this particular traffic criteria as shown in the above figure. There is 7% performance improvement as compared to round-robin scheme and Age based scheduler outperforms baseline router by 2X.

Figure 6.9 shows the simulation results for Priority-based scheduling using uniform random traffic from 0.01 (very low load) to 0.7 (high load) injection rates. The result is consistent with our expectation. It can be seen that, when the packet injection rate is high, the performance of priority based scheduler outperforms the round-robin scheme and as well as fair-sharing, monopolizing and baseline router. We can see, among all schemes, channel-stealing is the best under Priority based scheduling for priority class one for this particular traffic criteria as shown in the above figure. The Static-Priority based scheduler outperforms the baseline by 2X

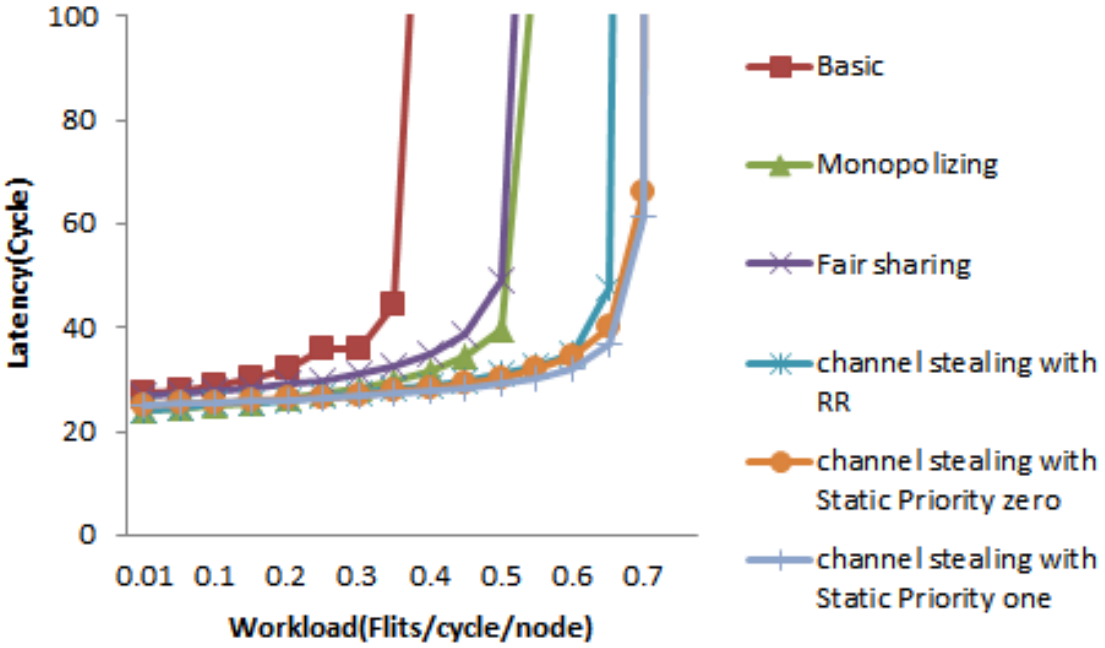

Figure 6.9: Performance of Static-Priority based scheduler under UR workload

and there is 7% performance improvement from round-robin in case of high priority packets.

Figure 6.10 shows the simulation results under BC traffic for all proposed scheduling algorithms with existing round-robin scheme in APCR. The results are produced using bit complement traffic from 0.01 (very low load) to 0.56 (high load) injection rates. It can be seen that, when the packet injection rate is low, the performance of all schedulers perform similar and also almost with same latency as in round-robin scheme. Here Static-Priority based scheduler does not work better than round-robin. However, we can see, among all schemes, channel-stealing under PF and Age scheduler policies perform best BC workload which can be observed from the above figure.

Figure 6.11 shows the simulation results under TP workload for all proposed scheduling algorithms with existing round-robin scheme in APCR. The results are

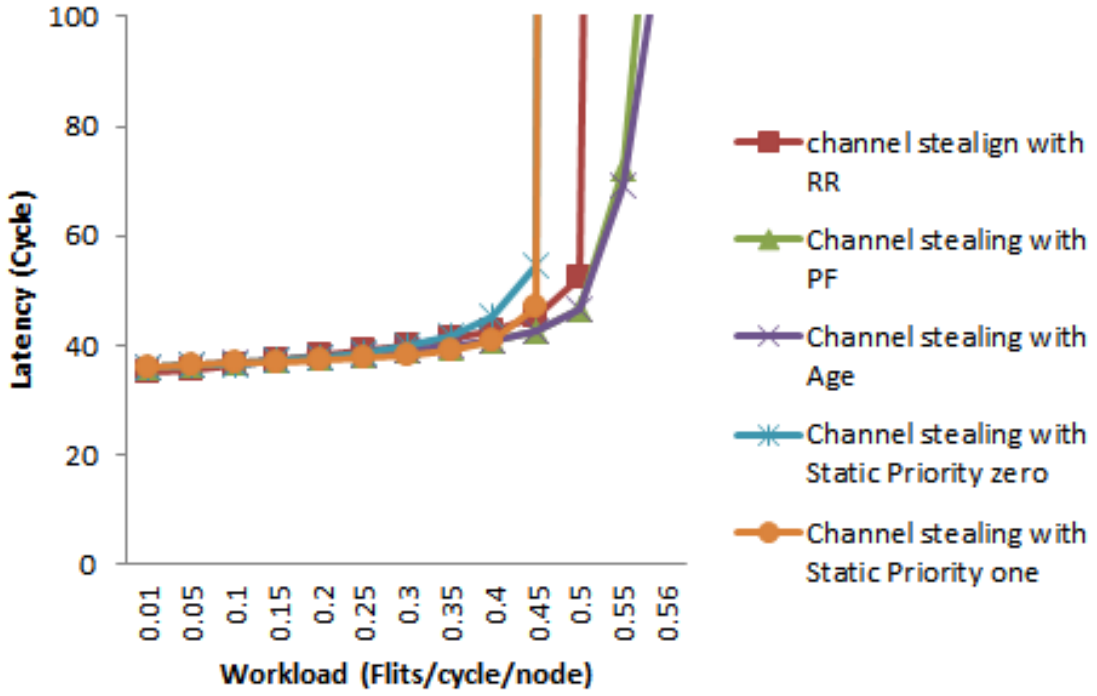

Figure 6.10: Comparison of RR, PF, Age and Static-Priority based scheduling policies under BC workload

produced using transpose traffic pattern from 0.01 (very low load) to 0.56 (high load) injection rates. We can see, among all schemes, channel-stealing under priority based scheduling performs not very well for this particular traffic criteria as referred in the above figure. All our new schedulers perform better than round-robin under TP workload.

Figure 6.12 shows the simulation results under UR traffic for all proposed scheduling algorithms with existing round-robin scheme in APCR. The results are produced using uniform random traffic from 0.01 (very low load) to 0.56 (high load) injection rates. The result is consistent with our expectation. It can be seen that, when the packet injection rate is low, the performance of all scheduler performs similar and also similar to round-robin scheme. Among all schemes, all the scheduling poli-

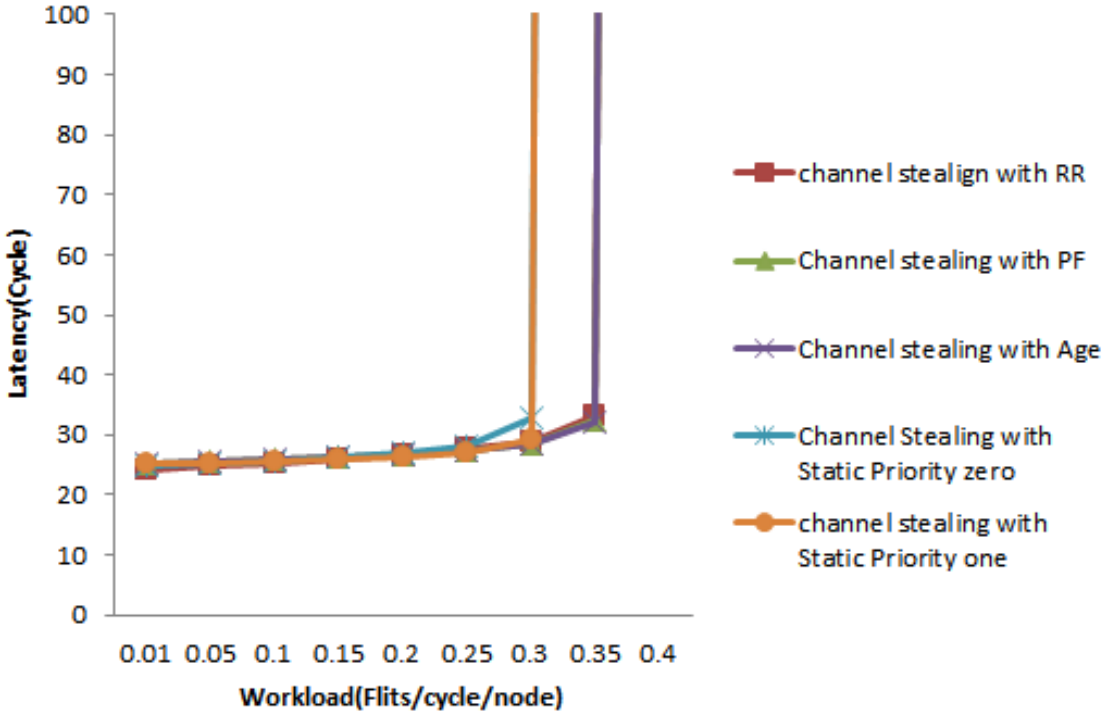

Figure 6.11: Comparison of RR, PF, Age and Static-Priority based scheduling policies Under TP workload

cies perform better than round-robin scheduler where as there is 10% performance improvement in case of PF scheduling.

6.2.1 Quality of Service

Evaluating satisfaction of quality of service requirement in any scheme involves with more than one parameters, such as fairness, power consumptions, area overhead, network throughput etc. For a given network size, latency delay may occur due to network congestion which is not acceptable for real-time applications. In this case, reserving some bandwidth for each real or non-real time services may guarantee a timely delivery of data packets among routers. This solution is able to overcome the latency problem, but it increases the power consumption and the cost of NoC design.

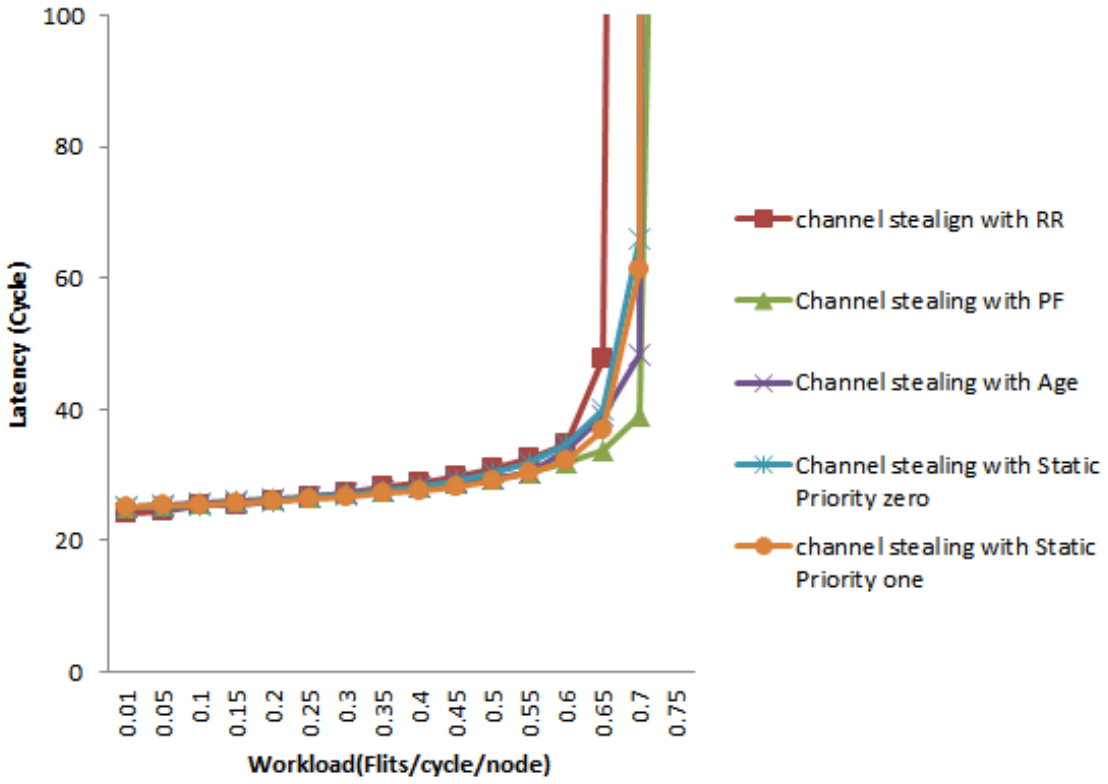

Figure 6.12: Comparison of RR, PF, Age and Static-Priority based scheduling policies Under UR workload

One of the cost effective solutions would be providing priority levels to the data traffic. In such systems, we are be able to successfully transmit the data packet for a real-time application scenario with less penalty. In all our three schemes, we have imposed priority to either each packets or flits depending upon the scheme we use. If we consider, Age based scheduler, it tried to minimize the packet latency delay which is a responsible scheme for providing best service to old packets prioritizing over new packets. Also, in Static-Priority based scheduling, we can impose high priority to real time traffics while giving low priority to non-real time traffics providing better service to real time traffics. So, providing best quality of service, being our one of the main motivation, we have implemented three priority based scheduling schemes that

successfully takes care of packets depending upon their age, priority and throughput.

6.2.2 Fairness

In our PF-scheduling, we have considered the past overall throughput of each VC or input ports. More is the throughput of the particular VC, more is the number of flits sent from that VC till last cycle which implies more is the number of times that VC is being selected in scheduling which is unfair to other VC that are waiting to send flits. So, we impose fairness here by providing this ratio based selection policy where higher is the number of flits the VC has sent the flits, less is the chance to send again in next cycle so that, it can let other VCs to send packets. Fairness can be latency fairness or throughput fairness. Here, we concern about latency fairness by supporting critical packets and diverse applications.

7. CONCLUSION

In this thesis, we have implemented three scheduling disciplines such as ratio based PF scheduler, randomly imposed Static-Priority scheduler, and Age based scheduling policy that eliminate the drawbacks of round-robin scheme when implemented in APCR model. Because, our policies not only provide the fairness, but also provide the best service to packets of either same or different applications. We provide a solution to achieve low latency for each packet in the network. PF scheduler maximizes the average throughput of the network while providing fairness to all VCs and input ports, where as Age based scheduler minimizes the individual packet latency. Again, Static-Priority based scheduler provides provision for supporting multiple application types with different time criticalities. All of these three policies have been implemented in the channel stealing scheme of APCR router. We studied and demonstrated each of the scheduling policy and compared their performance with existing round-robin for channel stealing. We also compared the performance difference among all new scheduling algorithm as well as round-robin scheduler under three synthetic traffic patterns such as bit complement, transpose and uniform random with different workload. It is seen that the PF scheduler outperforms the baseline router, round-robin in APCR under bit complement workload whereas Age based scheduler performs almost similar to PF-based scheduler in all three traffics. Again, under uniform random traffic, Static-Priority based scheduler performs better at high priority as compared to round-robin. All the three schedulers perform better than round-robin under TP workload. There is 12X performance improvement in both PF scheduler and Age based scheduler as compared to round-robin scheduler under BC traffic.

It is also observed that our Static-Priority based scheduler can be applied to diverse real time traffics. Extending our work to more realistic and complex settings will be an interesting aspect for our future study. Also, it can be enumerated to evaluate our proposed methods with more experiments considering other types of benchmark traffics. Implementing our scheduling policies on modern topologies like FBLY, MECS, MESHX etc will definitely be an interesting aspect for research.

REFERENCES

[1] J. Owens, W. Dally, R. Ho, S. Keckler, and L. Peh. Research challenges for on-chip interconnection networks. *IEEE Micro*, 27(5):96–108, 2007.

[2] J. Balfour and W. Dally. Design tradeoffs for tiled CMPs on-chip networks. In *Proceedings of the 20th Annual International Conference on Supercomputing, ICS'06*, pages 187–198. ACM, 2006.

[3] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proceedings of the 38th Annual Design Automation Conference, DAC'01*, pages 684–689. IEEE, June 2001.

[4] R. Mullins, A. West, and S. Moore. The design and implementation of a low-latency on-chip network. In *Proceedings of the 2006 Asia and South Pacific Design Automation Conference, ASP–DAC'06*, pages 164–169. IEEE Press, 2006.

[5] L. Wang, P. Kumar, K. Yum, and E. J. Kim. APCR: Adaptive physical channel regulator for on-chip interconnects. In *Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques, PACT'12*, pages 87–96. ACM, 2012.

[6] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *Proceedings of the Conference on Design, Automation and Test in Europe, DATE*, pages 250–256. IEEE, 2000.

[7] B. Daniel and W. Dally. Allocator implementations for network-on-chip routers. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC'09*, pages 1–12. ACM, 2009.

[8] S. Mukherjee, F. Silla, P. Bannon, J. Emer, S. Lang, and D. Webb. A comparative study of arbitration algorithms for the alpha 21364 pipelined router. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'02*, pages 223–234. ACM, 2002.

[9] I. Keslassy, M. Kodialam, T. V. Lakshman, and D. Stiliadis. On guaranteed smooth scheduling for input-queued switches. *IEEE/ACM Transactions on Networking*, 13(6):1364–1375, 2005.

[10] A. Raina and V. Muthukumar. Traffic aware scheduling algorithm for network on chip. In *Proceedings of the Sixth International Conference on Information Technology: New Generations, ITNG'09*, pages 877–882. IEEE, 2009.

[11] T. Ji, E. Athanasopoulou, and R. Srikant. Optimal scheduling policies in small generalized switches. In *Proceedings of the 28th Annual IEEE International Conference on Computer Communications, INFOCOM'09*, pages 2921–2925. IEEE, 2009.

[12] D. Andreasson and S. Kumar. Slack-time aware routing in NoC systems. In *Proceedings of IEEE International Symposium on Circuits and Systems, ISCAS'05*, volume 3, pages 2353–2356. IEEE, 2005.

[13] C. Kim, D. Burger, and S. W. Keckler. An adaptive, non-uniform cache structure for wire-delay dominated on-chip caches. In *Proceedings of the 10th*

International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS'02, pages 211–222. ACM, 2002.

[14] J.-M. Jou and Y.-L. Lee. An optimal round-robin arbiter design for NoC. *Journal of Information Science and Engineering*, 26(6):2047–2058, 2010.

[15] B. Towles and W. Dally. Guaranteed scheduling for switches with configuration overhead. *IEEE/ACM Transactions on Networking*, 11(5):835–847, 2003.

[16] M. Winter and G. P. Fettweis. A network-on-chip channel allocator for run-time task scheduling in multi-processor system-on-chips. In *Proceedings of the 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, DSD'08*, pages 133–140. IEEE, 2008.

[17] F. Guderian, E. Fischer, M. Winter, and G. Fettweis. Fair rate packet arbitration in network-on-chip. In *Proceedings of the 2011 IEEE International Symposium on Cloud Computing, SOCC'11*, pages 278–283. IEEE, 2011.

[18] D. Abts and D. Weisser. Age-based packet arbitration in large-radix k-ary n-cubes. In *Proceedings of the 2007 ACM/IEEE Conference on Supercomputing, SC'07*, pages 1–11. IEEE, 2007.

[19] K. H. Yum, E. J. Kim, and C Das. QoS provisioning in clusters: an investigation of router and NIC design. In *Proceedings of the 28th Annual International Symposium on Computer Architecture, ISCA'01*, pages 120–129. ACM, 2001.

[20] A. A. Chien and J. H. Kim. Rotating combined queuing (RCQ): bandwidth and latency guarantees in low-cost, high-performance networks. In *Proceedings*

of the 23rd Annual International Symposium on Computer Architecture, ISCA'96, pages 226–236. ACM, 1996.

[21] M. M. Lee, J. Kim, D. Abts, and J. W. Lee. Probabilistic distance-based arbitration: Providing equality of service for many-core CMP. In *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, Micro'07*, pages 509–519. IEEE, 2010.

[22] N. McKeown. The islip scheduling algorithm for input-queued switches. *IEEE Transactions on Networking*, 7(2):188–201, 1999.

[23] S. Balakrishnan and F. Ozguner. A priority-driven flow control mechanism for real-time traffic in multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems*, 9(7):664–678, 1998.

[24] S. Thoziyoor, N. Muralimanohar, J. H. Ahn, and N. P. Jouppi. Cacti 5.1. Technical report, Technical Report HPL-2008-20, HP Laboratories, Palo Alto, CA, April 2008.